Access 97 VBA
סדנת לימוד

הוראות מדויקות להתקנת התקליטור והפעלת התוכנות נמצאות בנספח

עורך ראשי: **יצחק עמיהוד**

בדיקה, עריכה מקצועית ועיצוב: **אניקה סואץ**

עיצוב עטיפה: **סטודיו מצגר**

שמות מסחריים

שמות המוצרים והשירותים המוזכרים בספר הינם שמות מסחריים רשומים של החברות שלהם. הוצאת הוד-עמי עשתה כמיטב יכולתה למסור מידע אודות השמות המסחריים המוזכרים בספר זה ולציין את שמות החברות, המוצרים והשירותים. שמות מסחריים רשומים (registered trademarks) המוזכרים בספר צוינו בהתאמה.

Access 97, Windows 95/98 הינם מוצרים רשומים של חברת Microsoft

לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד. ספר זה מיועד לגברים ונשים כאחד ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים/ות.

- **טלפון: 09-9564716**
- **פקס: 09-9571582**
- **דואר אלקטרוני: info@hod-ami.co.il**
- **אתר באינטרנט: www.hod-ami.co.il**

Access 97 VBA
סדנת לימוד

פרופ׳ אבא אנגלברג

Designed for Microsoft® Windows NT® Windows®98

הוצאת הוד-עמי
לספרי מחשבים

Access 97 VBA Workshop
By Prof. Aba Engelberg

Editor: **I. Amihud**

הוצאת הוד-עמי
לספרי מחשבים בע"מ
ת.ד. 6108 הרצליה 46160
טלפון: 09-9564716 פקס: 09-9571582
info@hod-ami.co.il

הודפס בישראל 1999

ISBN 965-361-205-0 מסת"ב

תוכן עניינים מקוצר

תוכן העניינים

5 שפת שאילתות סטנדרטית (SQL) 111

6 ניפוי תוכניות 167

7 חיבור לטפסים 171

נספח: התקליטור המצורף 187

קורא יקר !!!

ספר זה מציג את תהליכי ההפעלה
והכיוונון של מערכות ההפעלה
Windows 95 ו- Windows 98, ושל
התוכנה Access 97.

התוכנות מתאימות את המסכים
לתצורת החומרה והתוכנה
שמותקנות במחשב שלך.

על כן, ייתכן שהמסכים שתראה על
גבי המחשב שלך יהיו שונים במעט
מאלה המוצגים בספר.

14 Access 97 VBA - סדנת לימוד

ביבליוגרפיה

- Microsoft Access 97 for Windows Superguide, Miriam Liskin, Ziff-Davis Press, 1997

- Microsoft Access/Visual Basic Step by Step, Evan Callahan,1995, Microsoft Press

- Access 95 VBA Programming, Robert Smith and David Sussman, 1996, Wrox Press Ltd.

מבוא

VBA (Visual Basic) היא שפה שפותחה על ידי חברת Microsoft, במטרה לשמש כשפת עזר אחידה עבור היישומים השונים שמהווים את חבילת Microsoft Office 97. שפה זו קיימת במספר גרסאות שונות. כמוצר בודד, Visual Basic 6 היא אחת מהשפות הפופולריות ביותר, והיא מהווה מתחרה רצינית לשפת ++C, במיוחד בתחום התוכנות העסקיות, כאשר מהירות התגובה היא לא הגורם העיקרי. ביישומי Office: Word, Excel ו-Access מיושמת גרסת: **VBA** - **V**isual **B**asic for **A**pplication, לצד שימוש במאקרוס כאפשרות לשיפור ולחיזוק ביצועי היישומים. חשיבות VBA מודגשת בעיקר ב-Access, משום ששפה זו מנוצלת היטב לבניית מערכות סגורות, כאשר חלק מהביצועים הרצויים מחייבים שימוש בשפת עזר.

מבנה הספר

בארבעת הפרקים הראשונים של הספר מופיע הסבר ממצה ומקיף לעולם התכנות ב-VBA. שלושת הפרקים האחרונים, 5, 6 ו-7, מסבירים את השימוש העיקרי ב-VBA, כגיבוי וחיזוק לביצועי Access. פרק 5 כולל הצגה שלמה של שפת SQL, ושימושה ב-VBA ובמסדי נתונים של Access 97. פרק 6 עוסק בניפוי הקוד. פרק 7 מציג את השימוש ב-VBA באובייקטים השונים של Access 97, למשל טפסים ודוחות.

למי מיועד הספר

ספר זה מיועד לשני סוגי מפתחים.

- למפתח שחשיפתו הראשונה לעולם התכנות היא באמצעות Microsoft Office. למפתח זה מומלץ לקרוא בעיון את ארבעת הפרקים הראשונים, למלא אחר ההוראות הכתובות בספר, ולהריץ את התוכניות המודגמות. מומלץ גם לבצע חלק מהתרגילים שבסוף כל קטע, כדי להעמיק את הידע במושגים החדשים.
- למפתח בשפות אחרות, למשל פסקל או ++C, או אפילו Visual Basic (לא VBA), המעוניין לנצל את הידע הקודם שלו להשתלטות מהירה על VBA של Access. למפתח מסוג זה מומלץ לעבור ברפרוף על הפרקים הראשונים, ולהקדיש את עיקר תשומת הלב לפרקים 5, 6 ו-7. כן מומלץ לו להריץ מספר תוכניות בפרק הראשון, רק כדי להבטיח שהוא מבין כיצד מריצים תוכניות ב-VBA, ולהריץ מספר תוכניות בפרקים 2, 3 ו-4.

תוכניות וקטעי קוד המופיעים בספר

כל קטעי הקוד המופיעים בספר זה, לקוחים ממסד הנתונים Prog97.mdb שבתקליטור המצורף. כן צורף לתקליטור מסד הנתונים Order EntryH שנבנה באמצעות האשף, לשימוש בפרק 7. תוכל להשתמש במודולים בשלמותם, כדי לחסוך זמן, ולמנוע טעויות הקלדה במהלך העבודה עם הדוגמאות שבספר.

התקליטור המצורף

בתקליטור המצורף תמצא את תיקיית מסדי הנתונים של ספר זה - תיקיה בשם **59236** הנמצאת תחת התיקיה **Books**. בנוסף, תמצא בו את הקטלוג הצבעוני של **הוד-עמי**, מאמרים לדוגמה ותוכנות חופשיות. להרחבה על תכולת התקליטור ואופן השימוש בו, פנה לנספח.

מוסכמות המיושמות בספר זה

בספר זה יושמו מספר מוסכמות (רובן מוסכמות המקובלות במרבית ספרי ההוצאה):

- פקודות ושמות תפריטים יופיעו בטקסט **מודגש**.
- ביטויים ומונחים חדשים יופיעו גם הם בטקסט **מודגש**.
- צירופי מקשים המיועדים להקשה בו-זמנית, יופיעו כשסימן החיבור (+) מחבר בין קיצורי שמות המקשים, כך: Ctrl+X.
- קטעי קוד יופיעו על רקע אפור, כך:

```
Macro Text
```

כן יופיעו הסמלים הבאים, לציון:



רמז

סמל זה יופיע מימין לרמז או טיפ.



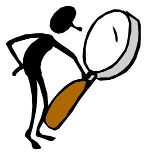
אזהרה:

סמל זה יופיע מימין לאזהרה.



תרגול:

סמל זה יופיע מימין לתרגילים שבגוף הפרק.



הערה:

סמל זה יופיע מימין להערה.

פרק 1

צעדים ראשונים

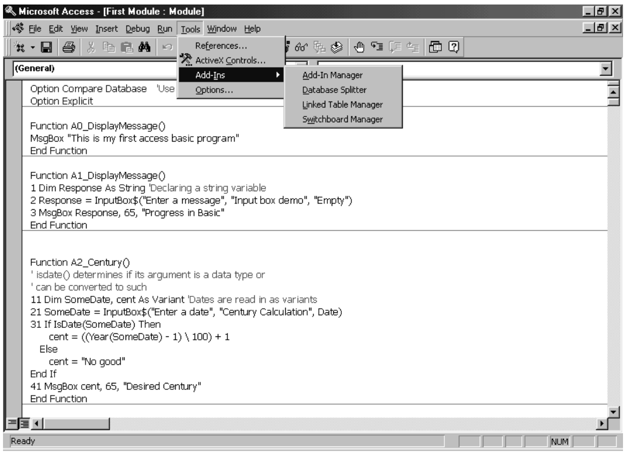
כדי להריץ תוכנית ב- Visual Basic של Access, אנו צריכים להיכנס ל-Access מ- Windows 95/98 באמצעות הצעדים הבאים :

1. לחץ על **התחל** (Start).
2. לחץ על **תוכניות** (Programs).
3. בחר **Microsoft Access** (הסדר תלוי בהתקנת Access, ובשינויים שערכת בתפריט **התחלה**), באופן חליפי, אם יש קיצור-דרך על שולחן העבודה, לחץ עליו.
4. בתיבת הדו-שיח שתופיע, בחר את **מסד נתונים ריק** (Empty Database), ולחץ על **אישור**.
5. נווט לתיקיה (Folder) המתאימה, הקלד את השם הרצוי, למשל Programs, ולחץ על **צור** (Create).

תיבת הדו-שיח שתופיע מהווה את התפריט הראשי של Access. בתיבה מספר כרטיסיות הנקראות : **טבלאות**, **שאילתות**, **טפסים**, **מודולים** וכולי. לענייננו רלוונטית הכרטיסיה **מודולים**. לחץ על **מודולים** ובחר **חדש** (New).



בחלון שייפתח יוצגו שורת התפריט וסרגל הכלים. תפריטים תוכל להפעיל על ידי לחיצה על שם התפריט בשורת התפריטים, או הקשת Alt בצירוף האות המודגשת בקו תחתי שבשם התפריט. בבחירת אפשרות תפריט, נפתח תפריט משנה (או תיבת דו-שיח). האפשרות הרצויה בתפריט המשנה נבחרת באופן זהה.



להלן התפריטים והאפשרויות הזמינות בהם:

- **קובץ** (File) - לפתיחת מסד נתונים אחר, לסגירת מסד נתונים זה, לשמירת מסד נתונים זה, לשמירת התוכניות במודול (או חלק מהם) כקובץ טקסט או HTML, להדפסת התוכניות (או חלק נבחר מהן) אשר במודול.
- **עריכה** (Edit) - להעתקה או העברת חלק מהטקסט ממקום למקום במודול זה, או למודול אחר, לחיפוש והחלפת טקסט, להכנסת טקסט פנימה או החוצה.
- **תצוגה** (View) - לקביעת חלון העבודה (רגיל או ניפוי), סרגלי הכלים, ולעזרה בסריקת אובייקטים.
- **הוספה** (Insert) - להוספת פרוצדורות (הליכים) או קטעי טקסט למודול הנוכחי, או להוספת מודולים חדשים.
- **אתר באגים** (Debug) - להידור וניפוי התוכניות.
- **הפעל** (Run) - להרצת התוכנית.
- **כלים** (Tools) - להוספת אפשרויות מסוימות ולקביעת התנהגות המערכת.
- **חלון** (Window) - לחלוקות שונות של החלונות כך שניתן לראות יותר מקטע אחד של מסד הנתונים בו-זמנית.
- **עזרה** (Help) - מספק עזרה בתחביר הפקודות ושימוש במסדי נתונים, כולל ב- Visual Basic.

סרגל הכלים שמופיע בחלון הפתיחה של המודול כולל סמלים לשימושים נפוצים מאוד. בהעברת הסמן על סמל, תופיע תווית שמסבירה את משמעותו.

מבנה הקוד

קוד Visual Basic מורכב משלושה חלקים :

- הצהרות (Declarations),
- פונקציות (Functions),
- תת-שגרות (Subroutines).

תצוגות שונות של המודול

אפשר לראות את הפרוצדורות במודול צמודות אחת לשנייה כאשר קו מפריד ביניהן, תצוגה זו נקראת **תצוגת מודול מלא** (Full Module View), וניתן לראות אותן כל אחת בנפרד, כשכל פרוצדורה ממלאת את כל המסך, תצוגה זו נקראת **תצוגת שיגרה** (Procedure View). כדי לעבור בין התצוגות השונות יש ללחוץ על הלחצנים שבתחתית ובשמאל המסך. כדי לגשת לשיגרה כלשהי, בחר את שמה מתיבת הרשימה שבראש ובימין החלון.

ניווט במודול בתצוגת המודול המלא

הניווט מתבצע בעזרת חיצי הגלילה, חיצי לוח המקשים, PgUp, PgDn, ^Home, ^End, או הזזת תיבת הגררה שבין חיצי הגלילה.

ניווט במודול בתצוגת שיגרה

כדי לנווט בתצוגה זו, הקש PgDn כשהסמן בשורה האחרונה במסך האחרון של השיגרה (אם השיגרה גדולה ממסך אחד), כדי להגיע לשיגרה הבאה. הקש PgUp כשהסמן בשורה הראשונה במסך הראשון של השיגרה כדי להגיע לשיגרה הקודמת.

בניית תוכניות



בפתיחת חלון הקוד נמצא בראשו שתי הצהרות. ההצהרה הראשונה: Option Compare Database, מתייחסת לסדר ההשוואות במחרוזות, והשנייה : Option Explicit, מכריחה אותנו להגדיר את סוג כל משתנה שנשתמש בו. שתיהן מקובלות עלינו, ואפשר להשאירן כמות שהן. נלחץ על החץ שליד הלחצן בצד שמאל של סרגל הכלים, בתפריט שיופיע נבחר **הליך** (Procedure), או בצורה חליפית, נבחר **הליך** (Procedure) מתפריט **הוספה**. בתיבת הדו-שיח שתופיע, נשאיר את ברירת המחדל **פונקציה** (Function) ונקרא לה A00. יכולנו גם להקליד בגוף חלון הקוד את המילים ()Function A00 ולהקיש Enter, מבלי ללחוץ על הלחצן להוספת הליך (שיגרה). המשמעות הספציפית של **פונקציה** תתבהר בעתיד. בשלב זה, די שנתייחס אליה כאל מסגרת שממלאים כדי לאפשר הרצת תוכניות. את לחצן

האפשרות עבור **טווח** (Range) נשאיר בברירת המחדל שלו - **ציבורית** (Public). משמעות אפשרויות אלו היא ששיגרה ציבורית ניתנת לקריאה מכל מודול במסד הנתונים, כאשר שיגרה **פרטית** (Private) נגישה רק מהמודול בו היא מוגדרת. שגרות פרטיות מאפשרות לנו לתת אותו שם לשגרות המופיעות במודולים שונים במסד הנתונים. כעת לחץ על **אישור**.

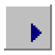
תוכנית A00: הפונקציה הראשונה שנבנה.

```
Public Function A00()
' Example A00. Using the computer as a calculator
'שימוש במחשב כמחשב כיס
MsgBox 6 + 5

End Function
```

הסבר לתוכנית A00**:** בשורה הראשונה מופיעים שם וכותרת הפונקציה: ()A00. הגרש בשורה השנייה מסמן שהטקסט שלאחריו הוא רק טקסט הערה, אם שורה שלמה היא הערה, אפשר לכתוב את המילה REM בתחילת המילה. ההערה מספרת לנו שבתוכנית הראשונה נשתמש במחשב כבמחשבון. כתבנו את ההערה באנגלית ובעברית. כדי לעבור לעברית, נלחץ על הלחצן **En** שבשורת המשימות ונבחר **He** (או, בצורה חליפית, נקיש Alt ו-Shift), כדי לחזור לאנגלית, נלחץ שוב על הלחצן שבשורת המשימות, או נקיש שוב את צירוף המקשים. הפקודה האופרטיבית היחידה בתוכנית היא: MsgBox 6 + 5, שמשמעותה היא שנדפיס את התוצאה בחלון המיועד לתשובה. בשורה האחרונה מופיעה הסיומת הסטנדרטית של הפונקציה (End Function).

הרצה והידור תוכניות



להרצת הפונקציה, לחץ על הלחצן **לך/המשך** (Go/Continue) שבסרגל הכלים, הקש F5, או בחר **לך/המשך** מתפריט **הפעל**.



כדי לוודא שלא שגינו, נבצע **הידור** (Compilation) שהוא פעולת תרגום. אם שגינו, Access תצביע על השגיאה. לדוגמה, שנה את MsgBox ל-MsagBox, והדר את הפונקציה על ידי בחירת **העבר קומפילציה של מודולים טעונים** (Compile Loaded Modules) מתוך תפריט **אתר באגים** (Debug) או לחיצה על הלחצן המתאים. נקבל הודעת שגיאה, וכאשר נאשר את קבלת ההודעה, המילה MsagBox תואר. נתקן את האיות ונבצע שוב הידור, ונראה ש-Access כבר אינה מתלוננת.

שמירה, הדפסה ויציאה מ-Access

אחרי שהרצנו את התוכנית הראשונה, נרצה לשמור אותה. סגור את המודול על ידי בחירת **סגור** (Close) מתפריט **קובץ** (File), או לחץ על X בצד ימין למעלה של חלון הקוד (דאג ללחוץ על X שבחלון הפנימי ולא בחיצוני, כדי לא לסגור את Access עצמה). בהודעה שתופיע לחץ **כן**, כלומר, אתה רוצה לשמור שינויים במודול1. הקלד את השם הרצוי, נקרא למודול **תכנות**. בעתיד, כדי להיכנס למודול הרצוי, בחלון מסד הנתונים, נאיר את המודול הרצוי ונלחץ על **עיצוב**. אם קיימות כבר מספר פרוצדורות במודול ותרצה להיכנס לפונקציה שבנינו, תלחץ על החץ שליד תיבת הרשימה **Procedure** שבימין חלון הקוד (העבר את מצביע העכבר מעל החץ כדי להציג תיאור כלי - ToolTip ובו שם התיבה), ותבחר את שם הפונקציה מהרשימה שתוצג.

אפשר גם לשמור את הפרוצדורה מבלי לסגור את המודול, על ידי לחיצה על הלחצן המתאים בסרגל הכלים, בחירת **שמור** (Save) מתפריט **קובץ** (File), או הקשת S^.



אפשר להדפיס שיגרה (שיגרה/פרוצדורה/הליך - לעניין ספר זה, כולם במשמעות Procedure) על ידי בחירת **הדפסה** (Print) מתפריט **קובץ** (File), הקשת P^, או לחיצה על הלחצן המתאים בסרגל הכלים. בשתי הדרכים הראשונות, ניתנות כמה אופציות להדפסה. הראשונה היא **הכל**, כלומר להדפיס את כל המודול, השנייה היא **עמודים**, כלומר הדפסת עמודים מסוימים, והשלישית היא להדפיס **רשומות נבחרות**, כלומר רק את הרשומות שנבחרו מראש לפני הפנייה להדפסה (שיטת הבחירה תוסבר בהמשך).

אפשר לצאת מ-Access על ידי בחירת **יציאה** (Exit) מתפריט **קובץ** (File), או לחיצה על X שמופיע בצד ימין למעלה של החלון החיצוני.

כללים למתן שמות

מודולים ופונקציות:

- אורך השם עד 64 תווים
- אסור לכלול את התווים "!" או "."
- אסור שהשם יהיה מילה שמורה.

דרישות נוספות לפונקציות:

- השם חייב להתחיל באות
- אסור לכלול רווחים בתוך השם
- השם חייב להיות ייחודי במודול.

משתנים

התוכנית הראשונה הדגימה כיצד המחשב פועל, אבל לא הצדיקה את השימוש במחשב, שהרי מחשב כיס מסוגל לעשות אותו דבר. עוצמת המחשב מבוססת על היכולת שלו לעבוד עם משתנים. משתנה הוא השם שנותנים למקום בזיכרון שיוכל להחזיק את הערך שנציב בו. בעזרת משתנים ניתן לכתוב תוכנית כללית שתעבוד בלי קשר לערך המדויק של הנתונים. רק בעת הרצת התוכנית, מציבים את הערך המדויק של כל משתנה, ומקבלים תוצאה מדויקת. לו היינו מזינים ערכים שונים היינו מקבלים תוצאה שונה, בלי שום צורך לשנות את פרטי התוכנית. שם משתנה חייב להתחיל באות, אורכו עד 40 תווים, והוא מורכב מאותיות, ספרות ו- "_" (בדרך כלל מפרידים בין מילים על ידי אותיות קטנות וגדולות, למשל MonthOfBirth, ולא Month_Of_Birth).

תוכנית A01:

```
Public Function A01(intNum1 As Integer, intNum2 As Integer)
' Example A01. Using the computer as a calculator
' שימוש במחשב כמחשב כיס
MsgBox "The sum of the numbers is: " & intNum1 + intNum2
End Function
```

הסבר לתוכנית A01: תוכנית זו דומה לתוכנית הראשונה, מלבד השינויים הבאים:

- בכותרת הפונקציה, בתוך הסוגריים נרשום את שמות המשתנים, שייקראו intNum1 ו-intNum2. הם מוגדרים כמשתנה מסוג **Integer**, זאת אומרת: מספר שלם. חייבים להגדיר למחשב את סוג המשתנה, כדי שהמחשב יידע כמה מקום להקצות לו. למשל, למספר שלם המחשב מקצה 2 בתים, שבהם אפשר להחזיק מספרים מ: 32,768- עד 32,767. בהמשך נראה שלמשתנים מסוגים שונים המחשב מקצה כמויות שונות של מקום. שם כל משתנה מסוג Integer מתחיל בקידומת int, כדי שכנראה אותו נזהה מייד את סוגו. כאשר כוללים שמות משתנים בכותרת הפונקציה, המשתנים נקראים **פרמטרים** (Parameters).
- בגוף התוכנית, במקום 5 + 6, שנקראים **קבועים** (Constants), נכתוב intNum1 + intNum2. פקודה זו גורמת לכך שנוכל למצוא את סכום כל שני ערכים שנזין במקום intNum1 ו-intNum2.
- בפלט שילבנו גם מלל, כדי שמשמעותו תהיה ברורה. את המלל הקפנו בגרשיים, והפרדנו בין המלל והתוצאה באמצעות הסימן &, סימן זה מחבר את המלל והתוצאה למחרוזת אחת. עשינו זאת משום שהפעולה MsgBox מסוגלת לפלוט רק מחרוזת אחת. שים לב שהכנסנו רווח בסוף המלל לפני הגרשיים, כדי שבפלט יופיע רווח בין סוף המלל והתוצאה המספרית.

אי אפשר להריץ תוכנית זו כפי שהרצנו את התוכנית הראשונה, משום שהתוכנית דורשת ערכים עבור שני המשתנים intNum1 ו-intNum2, והשיטה הראשונה אינה מאפשרת הזנת נתונים. במקום השיטה הראשונה, נבצע את הצעדים הבאים:

1. לחץ על הלחצן **חלון איתור באגים** (Debug Window), בחר אופציה זו מתפריט **תצוגה** או הקש ^G.



2. בחלון שמופיע, בחלק התחתון, כתוב: **A01 (24,56) ?** והקש Enter, כאשר הסמן הוא בסוף המשפט. כמובן, מותר לרשום כל שני מספרים בתנאי שסכומם נמצא בטווח המותר למספר שלם.

אופרטורים מתמטיים

בתוכנית הקודמת השתמשנו באופרטור מתמטי אחד, "+". ניתן להרכיב ביטויים אריתמטיים על ידי שילוב מספר משתנים או קבועים בעזרת אופרטורים מתמטיים. למשל, A + 7 הוא ביטוי אריתמטי המשלב את המשתנה A עם המספר השלם 7 בעזרת האופרטור "+". בביטויים אריתמטיים ניתן להשתמש באופרטורים שלהלן:

- סוגריים: ()
- פונקציות: יוסבר בהמשך
- העלאה בחזקה: ^
- שליליות: זאת אומרת - כאופרטור על אופרנד אחד, למשל A- או 12-
- כפל: *
- חילוק: /
- חילוק שלם: \ (זאת אומרת: התוצאה של 4 \ 7 היא 1 ואין מטפלים בשארית)
- שארית: MOD (זאת אומרת: התוצאה של 4 MOD 7 היא 3)
- חיבור: +
- חיסור: -

הסדר בו הצגנו את האופרטורים אינו מקרי. הוא משקף את סדר הקדימויות. דוגמה: 5 / 15 + 15 שווה ל-18 ולא ל-6, משום שקודם מבצעים פעולת חילוק ומקבלים 3, ואחר מוסיפים 15. אם נוסיף סוגריים, שהם בקדימות העליונה ונכתוב 5 / (15 + 15), הערך באמת יהיה 6, משום שקודם מבצעים את תוכן הסוגריים, דהיינו פעולת חיבור שתוצאתה 30, אחר מחלקים ב-5.

סוגי משתנים

כפי שהזכרנו, בעבודה עם משתנים מסוג Integer אנו מוגבלים למספרים ותוצאות שאינם חורגים מטווח המספרים השלמים. מה אם הבעיה מחייבת מספרים גדולים יותר? קיימים עוד 5 סוגי משתנים מספריים והם:

סוג	טווח	דוגמה עם קידומת נכונה	בית
Long	שלמים עד -/+ שני מיליארד	Dim lngSchum as Long	4
Single	ממשיים עד -/+ 3.4 בחזקת 38	Dim sngSchum as Single	4
Double	ממשיים עד -/+ 4.9 בחזקת 324	Dim dblSchum as Double	8
Currency	4 ספרות עשרוניות עד -/+ 922 טריליון	Dim curSchum as Currency	4
Byte	0-255	Dim bytSchum as Byte	1

נוכל עכשיו לכתוב תוכניות עם סוגים שונים של משתנים ונתונים.

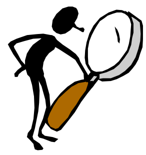
תוכנית A02: חישוב אורך וקטור.

```
Public Function A02(intX1 As Integer, intY1 As Integer,  _
                intX2 As Integer, intY2 As Integer)
' Example A02. Finding the length of a vector
'שימוש במחשב כמחשב
Dim sngLength As Single
sngLength = Sqr((intX1 - intX2) ^ 2 + (intY1 - intY2) ^ 2)
MsgBox "The length of the vector is: " & sngLength
End Function
```

בתוכנית A02 כללנו מספר חידושים והם:

- חלוקת פקודות ארוכות למספר שורות: בסוף השורה הוספנו לפחות רווח אחד, אחר הקלדנו את התו "_" והמשכנו את הפקודה בשורה הבאה. גם הערה אפשר להמשיך כך.
- לא כתבנו את כל המשתנים כפרמטרים. התוצאה, שתיקרא sngLength, הוגדרה כמשתנה בתוך התוכנית בעזרת **פקודת הצהרה** (Declaration Statement). כשכותבים את פקודת ההצהרה: Dim sngLength as Single, התוכנה מכירה את המשתנה sngLength כמשתנה מסוג Single. המילה Dim מודיעה שאנו עומדים להצהיר על משתנה, ומכתיבה למחשב להקצות לו מספיק מקום בזיכרון.
- הפקודה: sngLength = Sqr((intX1 - intX2) ^ 2 + (intY1 - intY2) ^ 2) נקראת **פקודת הצבה** (Assignment Statement), משום שמציבים את ערך הביטוי האריתמטי שמופיע בימין הפקודה במקום הערך הקודם של המשתנה שמופיע בצד שמאל. בצד שמאל של פקודת הצבה חייב תמיד להופיע שם משתנה.

- במקום להצהיר על משתנה בתוך התוכנית, יכולנו לייחס את התוצאה לשם הפונקציה, כמו שעשינו בתוכנית A03. שים לב להבדלים בין A02 ו-A03:
  - מוסיפים את המילים As Single בסוף הכותרת ב-A03. משמעות המילים היא שהתוצאה, שתאוכסן בשם הפונקציה, היא מסוג Single.
  - אין צורך להגדיר את המשתנה sngLength.
  - בפקודת ההצבה רושמים את שם הפונקציה בצד שמאל, והוא מחליף את המשתנה sngLength שבתוכנית A02.
  - פקודת MsgBox אינה נחוצה, שהרי התוצאה מוחזרת עם שם הפונקציה, וכבר ביקשנו אותה בפקודת הרצת התוכנית: **? A03(10,02,22,76)**.



הערה:

בנוסף לפונקציה Sqr להוצאת שורש, יש פונקציות מתמטיות נוספות, למשל Sin, Cos, ועוד. את פרטי פונקציות אלו, ופרטי פונקציות מתמטיות אחרות ניתן לראות על ידי לחיצה על הלחצן **סורק האובייקטים** (Object Browser), בחירת הערך **VBA** בתיבת הרשימה בצד שמאל למעלה של חלון סורק האובייקטים, בחירת **Math** מתוך החלונית השמאלית **Classes**, ובחירת הפונקציה הרצויה בחלונית הימנית. תיאור מפורט של הפונקציה יופיע בקטע התחתון של החלון.



תרגיל 1:

כתוב פונקציה שמקבלת כקלט את מספר שעות עבודה ושווי שעת עבודה של פועל, ופולטת את השכר המגיע לו.

תרגיל 2:

בתרגיל הקודם, נניח שמס הכנסה הוא x%, והמשכורת כוללת כל חודש החזר הוצאה נלווית עבור ביגוד, כאשר המס אינו חל על ביגוד. הרחב את הפונקציה בהתאם.

תרגיל 3:

כתוב פונקציה שמקבלת כקלט את גובה האדם באינצ'ים ופולטת את גובהו בסנטימטרים. זכור שאינץ' אחד שווה ל-2.54 סנטימטרים.

תרגיל 4:

על המרת שקלים לדולרים נגבים דמי עמלה. כתוב פונקציה שמקבלת את הנתונים הבאים: **sngShekalim** – מספר השקלים, **sngAchuz** – אחוז העמלה, בדרך כלל הוא נע בין 1/8% ל-1/2%, **sngShaar** – השער הנוכחי, כלומר מספר השקלים בדולר אחד. פלט הפונקציה יהיה מספר הדולרים שיתקבלו תמורת השקלים.



תרגיל 5:

כתוב פונקציה שקוראת שני מספרים: מידות מלבן. הפלט יהיה היקף, שטח, ואורך האלכסון. יש לכלול הערות, כדי להסביר את המשתנים ושיטת החישוב. הפלט יהיה משפט שיציג את משתני הקלט והפלט.

תרגיל 6:

כתוב פונקציה שקוראת שני מספרים: בסיס, ואורך היתר במשולש ישר זווית. הפלט יהיה גובה המשולש, סינוס, קוסינוס, וטנגס הזווית שבין היתר לבסיס. יש לכלול הערות, כדי להסביר את המשתנים ואת שיטת החישוב. מבנה הפלט יהיה משפט שיציג את משתני הקלט והפלט.

תרגיל 7:

כתוב תוכנית שמקבלת כקלט אורך, רוחב, ועומק של תיבה מלבנית. כתוב פונקציה שמחשבת את: שטח הדפנות מימין ומשמאל, שטח הגג והבסיס, שטח הדפנות מלפנים ומאחור, נפח התיבה. אם מ"ר של נייר אלומיניום עולה 30 שקל, כמה יעלה לעטוף את כל התיבה?

תרגיל 8:

תלמיד קיבל 3 ציונים במשך הסמסטר. כתוב תוכנית שתחשב את ממוצע הציונים, ההפרש שבין כל ציון והממוצע, הסטייה המוחלטת (כלומר הערך המוחלט של ההפרש) של כל ציון מהממוצע, ואת סטיית התקן של הציונים. סטיית תקן של 3 הציונים X1, X2, ו-X3 מוגדרת כך:

stiyat hateken = $\sqrt{\dfrac{3(X_1^2+X_2^2+X_3^2)}{3(3-1)}}$

תרגיל 9:

כתוב תוכנית שמקבלת כקלט מספר N, ומחשבת שורש, ריבוע, ומכפלה ב-5 וב-10 של 3 המספרים שלפני N, ו-2 המספרים שלאחריו.

תרגיל 10:

נתון רדיוס מעגל, כתוב שיגרה שמדפיסה את הקוטר, השטח, וההיקף.

תרגיל 11:

כתוב פונקציה שמקבלת כקלט טמפרטורה F בסולם פרנהייט, ומחשבת את המספר המתאים בסולם צלסיוס. נוסחת המעבר היא:

```
F = 9/5 * C + 32
```

תרגיל 12:

כתוב פונקציה שמקבלת מספר חשבון בבנק, יתרה קודמת, וכמות הכסף להפקדה או למשיכה. הפלט יהיה היתרה החדשה בחשבון.



תרגיל 13:

כתוב פונקציה שמחשבת את עלות הדלק לנסיעה, כאשר נתון מחיר ליטר דלק, מרחק הנסיעה, וקילומטרז' המכונית לליטר דלק.

תרגיל 14:

כתוב פונקציה שמקבלת משקל אדם בליברות, ומדפיסה את משקלו באונקיות, בקילוגרמים ובגרמים. לידיעתך, ליברה אחת שווה ל-16 אונקיות ול-0.45359 קילוגרמים.

שיטות נוספות לקלט ופלט בתוכניות

תוכנית A04: שיטות נוספות לקלט ופלט.

```
Public Function A04()
' Calculates the length of a phone call. Assume 24 hour clock.
 Dim intShaa As Integer
 Dim intDaka As Integer
 Dim intShniya As Integer
 Dim intOrech As Integer
 intShaa = InputBox$ ("באיזו שעה התחלת לדבר", "שעת התחלה", 0)
 intDaka = InputBox$ ("באיזו דקה התחלת לדבר", "דקת התחלה", 0)
 intShniya = InputBox$ ("באיזו שנייה התחלת לדבר", "שניית התחלה", 0)
 intOrech = InputBox$ ("כמה זמן שוחחת", "אורך השיחה", 0)
 intShniya = intShniya + intOrech
 intDaka = intDaka + intShniya \ 60
 intShniya = intShniya Mod 60
 intShaa = intShaa + intDaka \ 60
 intDaka = intDaka Mod 60
 intShaa = intShaa Mod 24

 Debug.Print " שעת סיום השיחה: " ; intShaa
 Debug.Print " דקת סיום השיחה: " ; intDaka
 Debug.Print " שניית סיום השיחה: " ; intShniya

End Function
```

הסבר לתוכנית A04: תוכנית זו קולטת את השעה, דקה ושנייה של תחילת שיחה, וגם את אורכה, ומחזירה את השעה, דקה ושנייה של סיום השיחה. אנו מתחילים עם פקודות הצהרה עבור המשתנים. את תוכן כל אחד נקלוט בעזרת פקודה חדשה: InputBox, שגורמת לכך שתיבת קלט תופיע על המסך ותאפשר קליטת ערך משתנה מסוג כלשהו. לפונקציה InputBox שלושה פרמטרים, הראשון הוא התוכן שיופיע בתיבה, השני הוא הכותרת והשלישי הוא ברירת המחדל. במקרה שלנו, קבענו ברירת

מחדל 0 לכל המשתנים, ונתנו לכל תיבה שאלה וכותרת מתאימה. חישבנו בעזרת פקודות הצבה המורכבות מביטויים אריתמטיים את זמן סיום השיחה, והוצאנו אותו כפלט. בדוגמה זו השתמשנו בפקודה חדשה לפלט, Debug.Print. פקודה זו מסוגלת להדפיס מספר שורות אחת אחרי השנייה, מבלי לדרוש את אישור המפעיל לאחר כל שורה, כמו שזה מתבצע בפקודת MsgBox. גם פה, שילבנו מלל עם הפלט כדי שמשמעותו תהיה ברורה, ודאגנו שיהיה רווח אחד בסוף המלל כדי להפריד בין המלל והתוצאה המספרית. הפעם, הפרדנו בין המלל והתוצאה בנקודה פסיק, סימן הגורם לשני הפלטים באותה שורה להופיע צמודים כתוצאה מפקודת Debug.Print. בעזרת הפקודה Debug.Print אנו יכולים להציג את כל הפלט בבת-אחת.

פקודת MsgBox - הרחבה

פקודת MsgBox שבה השתמשנו בתחילת הפרק, יכולה לקבל 3 פרמטרים, כפי שניתן לראות מתוכנית A05. תחביר הפקודה המורחבת נראה כך:

```
MsgBox "הפקודה עם כל 3 הפרמטרים", 65, "התקדמות ב-Access"
```

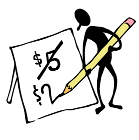
הפרמטר הראשון הוא המחרוזת שתוצג בתיבה, הפרמטר השני הוא קוד הצגה, והשלישי הוא כותרת התיבה. הפרמטר השני מבוסס על הקוד כדלהלן: 0 - לחצן אישור, 1 - לחצן אישור וביטול, 64 - סמל מידע (תמונת האות i). אם מחברים את הקודים 1 + 64, נקבל 65, ותוצג תיבת הודעה עם סמל מידע ולחצני **אישור** ו**ביטול**. קיימים קודים אחרים, אך כל סכום הוא ייחודי. כדי לראות את הקודים האחרים, סמן את המילה MsgBox, והקש F1, או בדרך חלופית: בתפריט **עזרה**, בחר **תוכן ואינדקס** (Contents and Index), לחץ על הכרטיסיה **אינדקס**, הקלד בתיבת הטקסט הריקה את המילה MsgBox, ולחץ Display או הקש Enter. שים לב, שעבור הפרמטר השני ניתן להשתמש בסכום של קבועים בעלי שמות המציינים את השפעתם. למשל, בדוגמה שלנו יכולנו לכתוב:

```
MsgBox "הפקודה עם כל 3 הפרמטרים", vbOKCancel + vbInformation, "התקדמות ב-Access"
```

הקבוע הראשון מחליף את הקבוע 1, והשני את הקבוע 64. הקידומת vb מרמזת על Visual Basic.

תוכנית A05: פקודת MsgBox.

```
Public Function A05()
MsgBox "הפקודה עם כל 3 הפרמטרים",  65, " התקדמות ב-Access "
End Function
```



תרגיל 15:

עבור על התרגילים הקודמים, ושפר את הקלט והפלט בעזרת השיטות החדשות שהוצגו.

עריכה

לפעמים, בעת הקלדת התוכנית, חלה שגיאת הקלדה שדורשת תיקון. לפעמים אפשר להעתיק קטע מתוכנית מסוימת לתוכנית הנוכחית ולחסוך זמן. לפעמים רוצים לחפש מילה מסוימת בתוכנית קיימת כדי לזכור כיצד לתכנת קטע כלשהו. את כל הדברים האלה ניתן לעשות בעזרת העורך של Access VBA.

- **מחיקת תווים**: אפשר למחוק את התו שלפני הסמן על ידי הקשת Del (שקיים פעמיים במרבית המקלדות). אפשר למחוק את התו האחרון שלפני הסמן על ידי הקשת BackSpace (בשורה העליונה של המקלדת, עם חץ הפונה שמאלה). כמובן אפשר לחזור על הפעולה מספר פעמים, כדי למחוק מספר תווים אחד אחרי השני.
- **בחירת קטע**: לפני שמוחקים, מעתיקים, או מעבירים קטע, יש צורך לציין באיזה קטע מדובר. פעולה זו נקראת בחירה או Selection, והיא נעשית באחת מהדרכים הבאות:
  - **מילה**: לחץ לחיצה כפולה על העכבר.
  - **שורה**: לחץ בצד השורה, או גרור את העכבר על פני כל השורה. המשך לגרור על השורות האחרות.
  - **קטע כלשהו**: לחץ על הלחצן השמאלי של העכבר בתחילת הקטע, הקש Shift, ואחר לחץ על לחצן העכבר בסוף קטע הבחירה.
  - **כל המודול**: בחר Select All מתפריט עריכה, או הקש A^.
- **מחיקת קטע**: אחרי סימון הקטע, מוחקים אותו על ידי הקשת Del.
- **העברה והעתקה**: העברת קטע מוגדרת כהעברת שטח נבחר ממקום אחד למקום אחר באותה שיגרה או בשיגרה אחרת. הקטע נמחק ממקומו המקורי, ומועבר למקום החדש. בהעתקה, הקטע מועבר למקומו החדש, אך גם נשאר במקומו המקורי. העברה מורכבת משתי הפעולות **גזירה והדבקה**, העתקה מורכבת משתי הפעולות **העתקה והדבקה**.

גזירה: העתקת הקטע מחלון העבודה ללוח של Windows המכונה ClipBoard, ומחיקתו ממקומו המקורי. הפעולה מתבצעת (אחרי בחירת קטע) על ידי: לחיצה בלחצן המתאים בסרגל הכלים, הקשת X^, או בחירת **גזור** (Cut) מתפריט **עריכה** (Edit). (הסימן ^ מציין הקשה על מקש Ctrl בשורה התחתונה של לוח המקשים, ואם כן X^, משמעותה הקשת X בזמן שמקש Ctrl לחוץ גם הוא).

העתקה: העתקת הקטע מחלון העבודה ללוח מבלי למחוק אותו ממקומו המקורי. הפעולה מתבצעת (אחרי בחירת קטע) על ידי: לחיצה בלחצן המתאים בסרגל הכלים, הקשת C^ או בחירת **העתק** (Copy) מתפריט **עריכה** (Edit).

הדבקה: העתקת קטע מהלוח לחלון העבודה. הפעולה מתבצעת על ידי: לחיצה בלחצן המתאים בסרגל הכלים, הקשת V^ או בחירת **הדבק** (Copy) מתפריט **עריכה** (Edit).

בנוסף לדרכים שהוזכרו, ניתן להעביר קטע על ידי בחירת הטקסט ואחר גרירתו למקום הרצוי. ניתן גם להעתיק קטע על ידי בחירת הטקסט, ואחר גרירתו למקום הרצוי בזמן שמקש Ctrl לחוץ.

- **תיקון טעות**: אם טעית בזמן מחיקה, העתקה, העברה, או הדבקה, אפשר להתחרט על ידי: הקשת Z^, בחירת **בטל** (Undo) מתפריט **עריכה**, או לחיצה על לחצן **בטל** (Undo). כדי להחזיר מה שביטלת, אפשר לבחור: **בצע שוב** (Redo) מתפריט **עריכה**, או ללחוץ על הלחצן **בצע שוב** (Redo). אפשר להפעיל **בטל** כמה פעמים שרוצים כדי לבטל אחת אחת את כל הפעולות שעשית מאז תחילת העריכה או ההקלדה, ואפשר גם לשחזר אותן אחת אחת.
- **חיפוש**: כדי למצוא מילה מסוימת בשיגרה או במודול, הקש F^ או בחר **חיפוש** (Find) מתפריט **עריכה** (Edit). אחרי החיפוש הראשון, אפשר להמשיך את החיפוש על ידי הקשת F3.
- **החלפה**: כדי למצוא מילה מסוימת, ולהחליף אותה במילה אחרת: הקש H^, בחר **החלפה** (Replace) מתפריט **עריכה** (Edit). אחרי ההחלפה הראשונה, אפשר להמשיך את ההחלפות אחת אחת על ידי לחיצה ב**החלף** (Replace), או לוותר על החלפה מסוימת וללחוץ את הלחצן **חפש את הבא** (Find Next). אפשר גם להחליף את כל מופעי המילה על ידי בחירת **החלף הכל** (Replace All).

מחרוזות

משתנה אינו חייב להחזיק רק ערכים מספריים. הוא יכול גם להחזיק שמות, סוגי דם או כל ערך מילולי אחר. משתנה זה הוא מסוג **מחרוזת** (String), והוא יכול להחזיק עד 65,535 תווי (בתים) טקסט. מחרוזת מוצגת תמיד בתוך גרשיים. כבר שילבנו מחרוזות בתיבת הקלט InputBox וגם בפלט בפקודות Debug.Print ו-MsgBox.

קליטת מחרוזות

נבנה עוד פונקציה. קודם נעתיק את התוכנית הקודמת על ידי העברת הסמן עליה והקשת C^. נפתח פונקציה חדשה, ונקרא לה A06. כאשר הפונקציה נפתחת, נקיש V^ והפונקציה המקורית תופיע. בנקודה זו, נתחיל לשנותה, עד שתוכנה יהיה כבתוכנית A06.

תוכנית A06: התוכנית השלמה

```
1 Dim Response as String ' הגדרת משתנה המחזיק מחרוזת
2 Response = InputBox ("הזן מסר","תיבת קלט למחרוזת","ריק")
3 MsgBox Response, 65, " התקדמות ב-Access "
```

הסבר: בשורה הראשונה, Dim מציין שאנו עומדים להגדיר משתנה בשם Response, ומסוג String (מחרוזת). אורך המחרוזת אינו מוגדר, והמשתנה יקבל ערך כתוצאה

ממילוי המחרוזת במספר מסוים של תווים. כדי שאורך המחרוזת יהיה קבוע, כותבים String * n, כש-n הוא האורך הרצוי.

המספר בתחילת השורה אינו הכרחי והוא נקרא **מספר שורה** (Line Number). אפשר גם להוסיף **תווית** (Label), שהיא שם ולאחריו נקודתיים.

בשורה השנייה, קלטנו ערך בעזרת InputBox, והכנסנו אותו למשתנה Response. בשורה השלישית, הוצאנו כפלט את התוכן שהזנו למחשב כתוצאה מהשורה השנייה.

אופרטורים למחרוזות

האופרטור היחיד שיש לו משמעות בהקשר של מחרוזות הוא פעולת + (או &) המאפשר קישור (Concatenation) של מחרוזות. יתרון השימוש באופרטור & שהוא אינו דו-משמעי, שהרי אין לו משמעות כלשהי בחיבור אריתמטי, רק בקישור מחרוזות. פעולות נוספות מופעלות על מחרוזות בעזרת פונקציות.

תוכנית A07: שימוש במחרוזת.

```
Public Function A07()

Dim Machrozet As String
Dim start As String
Dim Sof As String
Dim Emza As String
Dim Makom As Integer, tozaa As Integer

A: MsgBox String$(10, "A")
B: MsgBox Asc("A")
C: MsgBox Chr$(65) + "BCD"

D: Machrozet = "I am a good boy"
E: start = Left$(Machrozet, 4)
F: Sof = Right$(Machrozet, 3)
G: Emza = Mid$(Machrozet, 8, 4) 'start at 8 length of 4
Rem to separate lines, add carriage return (13 in Ascii)
Rem and linefeed (10 in Ascii)
H: MsgBox "The beginning is: " & start & Chr$(13) & Chr$(10) _
" & The end is: " & Sof & Chr$(13) & Chr$(10) _
" & The middle is:" & Emza
Rem mid$ may be used not only to extract a substring, but also to insert data in _
 a specific location in the substring, in which case it appears on the left of an equation
I: Mid$(Machrozet, 8, 4) = "bad "
J: MsgBox "The changed string is: " + Machrozet

' The function Len determines the length of a string.
K: MsgBox "The length of the new string is: " & Len(Machrozet)
```

```
' use Instr for finding the location of a substring
' inside a given string

L: Makom = InStr(Machrozet, "boy")
M: MsgBox "The location is: " & Makom

' a second form of Instr which says from which location to start
N: Makom = InStr(3, Machrozet, "I") 'a starting location has been
' added. Since 3 is after I the result will be 0 because he
' won't find it.
O: MsgBox "The location is: " & Makom

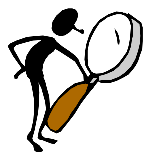
P: Makom = InStr(1, Machrozet, "Boy", 0) ' this is a second format
' for the function. Here the starting location is required
' The 0 at the end says the comparison is case sensitive, so
' here he won't find "Boy". Had we written 1 instead of 0, the
' comparison would not have been case sensitive.
Q: MsgBox "The location is: " + Str$(Makom)
 'changing to uppercase
R: Machrozet = UCase$("       Good Day"      )
S: MsgBox "The Left trim is: " & "start" & LTrim$(Machrozet) & "end" _
 &         Chr$(13) & Chr$(10) _
" &         The right trim is: " & RTrim$(Machrozet) & " end" _
 &         Chr$(13) & Chr$(10) _
" &   The total trim is: " & "start" & Trim$(Machrozet) & "end"
t: Machrozet = "אני ילד טוב ירושלים"
U: start = Left$(Machrozet, 3)
V: Sof = Right$(Machrozet, 7)
W: Emza = Mid$(Machrozet, 5, 7) 'start at 8 length of 4
X: MsgBox "The beginning is: " & start & Chr$(13) & Chr$(10) _
" &         The end is: " & Sof & Chr$(13) & Chr$(10) _
" &         The middle is:" & Emza


End Function
```

הסבר:

- **A**: הפונקציה String$(n,Char) בונה מחרוזת באורך n המורכבת מ-n הופעות של התו Char. פה בנינו מחרוזת המורכבת מ-10 הופעות של האות A. פונקציה דומה היא: Space$ (n) שבונה מחרוזת של n רווחים.
- **B**: לכל תו יש ייצוג מספרי פנימי במחשב. שם הקוד הוא קוד Ascii. הפונקציה Asc(Char) מציגה את הקוד עבור Char. פה הצגנו את הקוד עבור האות A שהוא 65 (אך הקוד עבור a הוא 97).

- **C**: הפונקציה $Chr היא ההופכית של הפונקציה Asc, כלומר, נתון קוד Ascii של תו כלשהו, והפונקציה מחזירה את התו עצמו. הארגומנט 65 יחזיר את התוצאה A, כפי שכבר ראינו בסעיף הקודם. בנוסף, קישרנו את המחרוזת BCD, כך שהתוצאה הסופית תהיה הצגת ABCD.
- **D-H**: מגדירים מחרוזת בשם Machrozet. הפונקציה Left$(Machrozet,4) שולפת את 4 האותיות השמאליות ביותר במחרוזת (I am), הפונקציה Right$(Machrozet,3) שולפת 3 אותיות ימניות במחרוזת (boy), והפונקציה Mid$(Machrozet,8,4) שולפת את 4 האותיות האמצעיות החל ממקום 8 (good). הכוונה היא שבעת ההצגה, ההתחלה, האמצע והסוף יופיעו בשורות נפרדות. כדי לגרום לכך, מוסיפים את שני התווים המיוחדים - Carriage Return ו-LineFeed. היות ואי אפשר להזין תווים אלה באמצעות לוח המקשים, מוסיפים את קוד Ascii שלהם למחרוזת להדפסה.
- **I-J**: הפונקציה Mid$(Machrozet,8,4) יכולה להופיע לא רק בצד ימין של המשוואה, כדי לשלוף תווים ממקום מסוים במחרוזת, אלא גם בצד שמאל, וזאת כדי להחליף את 4 התווים החל ממקום 8, בתווים שמופיעים בצד הימני של המשוואה, במקרה שלנו האותיות bad. שים לב שמספר התווים המחליפים (3 פה) לא חייב להיות שווה למספר האותיות המוחלפים (4 פה).
- **K**: הפונקציה LEN מחזירה את אורך המחרוזת כמספר שלם, שמשלבים עם מחרוזת התיאור.



הערה:

הפונקציה $Str הופכת ערך מספרי למחרוזת, ושמה רווח לפניו. הפונקציה CStr עושה אותו דבר, אך מבלי להוסיף רווח. פונקציות אלו נחוצות עבור פונקציות שמסוגלות לעבוד רק עם מחרוזות.

- **L-M**: עד עכשיו, שלפנו תווים ממקום מסוים במחרוזת. הפונקציה Instr(Machrozet1,Machrozet2) יודעת כיצד לחפש Machrozet2 (פה boy) בתוך Machrozet1. התשובה היא ערך מספרי שאומר באיזה מקום במחרוזת הראשונה נמצא התו הראשון במחרוזת השנייה (התשובה היא 0 אם המחרוזת השנייה לא נמצאת כלל במחרוזת הראשונה).
- **N-O**: הרחבה של Instr: Instr (n,Machrozet1,Machrozet2). הוספנו פה את הפרמטר n שמסמן מספר שלם, ומשמעותו היא שהחיפוש יתחיל רק ממקום מספר n במחרוזת.
- **P-Q**: עוד הרחבה של Instr: Instr (n,Machrozet1,Machrozet2,k). הפרמטר הרביעי k מוחלף על ידי 0 אם מבחינים בין אותיות לועזיות גדולות וקטנות. אם לא מקפידים, k שווה ל-1 (ברירת המחדל).
- **R**: הפונקציה Ucase (Machrozet) הופכת את כל התווים לאותיות רישיות.

- **S** : הפונקציה Ltrim (Machrozet) מורידה רווחים מובילים בצד השמאלי של מחרוזת, כאשר הפונקציה Rtrim (Machrozet) מורידה רווחים מהצד הימני, והפונקציה Trim (Machrozet) מורידה רווחים משני הצדדים.
- **T-X** : את הפונקציות האלו ראינו כבר במחרוזות המורכבות מאותיות לועזיות. כשהמחרוזות מורכבות מאותיות עבריות, למרות שהאותיות מופיעות מימין לשמאל, הן מאוחסנות משמאל לימין, ולכן הפונקציה Left$(Machrozet, 3) מחזירה את המילה "אני" שמופיעה בצד ימין (אך מאוחסנת בשמאל) ולא את האותיות "לים", וכדומה לגבי Right$(Machrozet, 7), שמחזירה את המילה "ירושלים" שמופיעה בצד שמאל אך מאוחסנת בצד ימין.



תרגיל 16:

כתוב שיגרה שקוראת שם אדם, גובהו בס"מ, ומשקלו בקילוגרמים. השיגרה מדפיסה את הנתונים שקיבלה, ובנוסף מחשבת ומדפיסה את הממוצע של מספר הקילוגרמים לכל מטר גובה, עם כותרת מתאימה.

תרגיל 17:

כתוב פונקציה שמקבלת כקלט מחרוזת שמכילה את הנתונים הבאים: שם פרטי, שם אמצעי, שם משפחה ושם חיבה, אשר מופרדים זה מזה ברווח אחד או יותר. הוצא את הפלטים הבאים:

A : שם פרטי

B : שם אמצעי

C : שם משפחה

D : שם פרטי ושם משפחה

E : שם חיבה ושם המשפחה

F : שם פרטי, אמצעי ושם משפחה

G : שם משפחה, פסיק ושם פרטי

משתנים בוליאניים

לעיתים יש צורך להעריך את היחס בין שני מספרים ולפעול בהתאם לתוצאה. למשל, מה התשובה שתצפה לקבל לשאלה: "האם A שווה ל-B?" האפשרויות הן שהמספרים שווים, או שאינם שווים. אפשר גם להשיב: **True** (נכון) או **False** (שגוי), בהתאמה. ב- Access VBA שומרים תשובה כזו ב**משתנה בוליאני** (Boolean Variable), אשר מקבל אחד מן הערכים שהזכרנו, **True** או **False**. יש גם ערך מספרי הקשור ל-False והוא 0, כאשר כל ערך אחר קשור ל-True.

תוכנית A08: משתנים בוליאניים.

```
Public Function A08(intNum1 As Integer, intNum2 As Integer)

' Declare boolean variables
 Dim booSmall As Boolean, booEqual As Boolean, booBig As Boolean
 booSmall = intNum1 < intNum2
 booEqual = intNum1 = intNum2
 booBig = intNum1 > intNum2
Debug.Print "The first number is smaller than the second is: "; booSmall
Debug.Print "The first number equals the second is: "; booEqual
Debug.Print "The first number is greater than the second is: "; booBig
booSmall = True < False
Debug.Print booSmall
booSmall = True > False
Debug.Print booSmall

End Function
```

הסבר: הגדרנו שלושה משתנים בוליאניים. באמצעות פקודת השמה משתנה בוליאני יכול לקבל את הערכים השמורים **True** או **False**. אפשר גם להציב תנאי בצד הימני של המשוואה. גם תנאי מקבל את הערכים True או False בלבד, והערך המתאים מועבר למשתנה הבוליאני באמצעות פקודת ההשמה. המחשב יכול לחשב **תנאי פשוט** (Simple Condition) הבנוי כך: **ביטוי אריתמטי**, **יחס**, **ביטוי אריתמטי**. היחסים האפשריים הם:

- = שוויון
- <> אי-שוויון
- > גדול מ-
- >= גדול או שווה ל-
- < קטן מ-
- <= קטן או שווה ל-

התנאי צריך להיות כזה שאפשר לקבוע את נכונותו. כלומר, הערכים המשווים צריכים להיות מאותו סוג. אי אפשר להשוות משתנה בוליאני שערכו **True** עם משתנה ממשי שערכו 10.97. מאידך, אפשר להשוות שני משתנים בוליאניים. הערך **True** נחשב כקודם ("קטן") לערך **False** (הערך המיוחס על ידי המחשב ל-**False** הוא 0, כאשר הערך המיוחס ל-**True** הוא 1-). ביטויים אריתמטיים אשר מהווים חלק מתנאי יכולים להכיל קבוע או משתנה אחד, או להיות מורכבים יותר, כמו למשל: Lachatz * (Chom - 4 * PI) <> Nefach - Lachatz. אפשר לשלב שני משתנים, קבועים או ביטויים אריתמטיים על ידי אופרטורים אריתמטיים, כדי לקבל ביטויים

אריתמטיים מורכבים. כך גם אפשר לשלב יחד תנאים פשוטים בעזרת האופרטורים הבוליאניים **AND**, **OR**, ו-**NOT** כדי לקבל **תנאים מורכבים (Complex Conditions).**
הערך המיוחס לשני תנאים פשוטים המשולבים על ידי התנאי AND יהיה True רק אם כל אחד משני התנאים יהיה True. מאידך, הערך המיוחס לשני תנאים המשולבים על ידי OR יהיה False רק אם כל אחד משני התנאים הוא False. האופרטור NOT פועל על תנאי אחד בלבד והופך את הערך שלו מ-True ל-False, ולהיפך. האופרטור **XOR** מחזיר ערך True אם אחד משני הביטויים הוא True, כאשר האופרטור **EQV** מחזיר ערך True אם לשני הביטויים הבוליאניים שהוא פועל עליהם יש אותו ערך (True או False). נניח שהכנסנו את הערכים של שני תנאים שונים למשתנים booP ו-booQ.
הטבלה שבהמשך, אשר נקראת **טבלת אמת** (Truth Table) מסכמת את השפעת האופרטורים השונים על המשתנים האלה.

P	Q	P AND Q	P OR Q	NOT P
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

תוכנית A09: התוכנית שיצרה את טבלת האמת.

```
Public Function A09()
' Declare boolean variables
Dim booP As Boolean, booQ As Boolean
Debug.Print "NOT P", "P OR Q", "P AND Q", "Q", "P"
Debug.Print
booP = True: booQ = True
Debug.Print booP, booQ, booP And booQ, booP Or booQ, Not booP
booP = True: booQ = False
Debug.Print booP, booQ, booP And booQ, booP Or booQ, Not booP
booP = False: booQ = True
Debug.Print booP, booQ, booP And booQ, booP Or booQ, Not booP
booP = False: booQ = False
Debug.Print booP, booQ, booP And booQ, booP Or booQ, Not booP
End Function
```

הסבר: נניח ש-A שווה ל-5, B שווה ל-10, ו-C שווה ל-2. אזי הביטוי:

```
(A < 7*C) AND (B+7 < A+C)
```

מהווה תנאי מורכב. ערך התנאי הפשוט השמאלי הוא True, מפני ש-A שערכו 5, באמת קטן מ- 7*C שערכו 14. ערך התנאי הפשוט השני: A+C > B+7 הוא False, מפני שערך B+7 הוא 17, שאינו קטן מ-A+C שערכו 7. מאחר והתנאי השמאלי True

והתנאי הימני False והם משולבים על ידי AND, ערך התנאי המורכב הוא False.
ביטוי אריתמטי הוא שילוב של אחד או יותר קבועים או משתנים מספריים באמצעות אופרטור אריתמטי. בדרך דומה ניתן להגדיר **ביטוי בוליאני** (Boolean Expression) כשילוב של תנאי אחד או יותר, ושל משתנים או קבועים בוליאניים (הכוונה למילים TRUE או FALSE), באמצעות אופרטורים בוליאניים. ערך ביטוי אריתמטי יכול להיות כל ערך בתחום המספרים שהמחשב יכול להחזיק. לעומתו, ערך ביטוי בוליאני מוגבל לשני ערכים בלבד, True או False. על אף התחום המצומצם של ערכו הסופי (True או False), ביטוי בוליאני יכול להכיל יותר אופרטורים מאשר ביטוי אריתמטי, שהרי תנאי מכיל ביטויים אריתמטיים (עם אופרטורים אריתמטיים) וגם אופרטורים של יחס. בביטוי בוליאני, אפשר לשלב יחד מספר תנאים על ידי אופרטורים בוליאניים. בשל המספר הרב של אופרטורים אשר אפשר לכלול בביטוי בוליאני, יש צורך להוסיף על חוקי הקדימות המקובלים. הסדר הכללי הוא שלאופרטורים אריתמטיים יש קדימות הכי גבוהה, אחר באים האורפטורים של יחס, ולבסוף האופרטורים הבוליאניים. סדר הקדימות של האופרטורים הבוליאניים, בינם לבין עצמם, הוא כדלהלן:

- NOT
- AND
- OR
- XOR
- EQV

עכשיו נוכל לראות שבדוגמת התנאי המורכב שהוצגה קודם :

```
(A < 7*C) AND (B+7 < A+C)
```

הסוגריים היו מיותרים, משום שהאופרטורים הלוגיים היו מופעלים בכל מקרה רק לאחר האופרטורים האריתמטיים.



תרגיל 18:

כתוב פונקציה שקוראת שני זוגות מספרים שלמים ומוציאה פלט של True או False עבור כל אחד מהתנאים הבאים:

- האם שני הזוגות זהים?
- האם האיבר הראשון והאיבר השני שווים בשני הזוגות?
- האם האיבר הראשון בזוג הראשון שווה לאיבר השני בזוג השני, או האם האיבר השני בזוג הראשון שווה לאיבר הראשון בזוג השני?



תרגיל 19:

חברת אל-על רוצה לגייס אנשי ביטחון מקרב תלמידי האוניברסיטה. הדרישות שלה הן כדלהלן:

- זכר
- גמר 85 נקודות לקראת התואר
- גובה למעלה מ-1.8 מטר
- משקל מעל ל-70 קילוגרם.

הקלט מורכב מ-5 נתונים:

- מספר סטודנט
- מין (זכר או נקבה)
- מספר נקודות שסיים באוניברסיטה
- גובה
- משקל

כתוב פונקציה שמוציאה כפלט את המשפט הבא:

סטודנט NNN מתאים להיות איש בטחון: XXXX (True או False)

תרגיל 20:

בשאלה הקודמת, מוכנים לקלוט גם נשים העונות לדרישות הבאות:

- נקבה
- גמרה 80 נקודות לקראת התואר
- גובה למעלה מ-1.7 מטר
- משקל מעל ל-55 קילוגרם.

מבנה הקלט כפי שתואר בשאלה הקודמת. כתוב פונקציה שמוציאה כפלט את המשפט הבא:

סטודנטית NNN מתאימה להיות אשת ביטחון: XXXX (True או False)

תרגיל 21:

חברה להפצת ספרים מקבלת הזמנות ושולחת ספרים. לקוח קבוע משלם רק כאשר הוא מקבל את הספרים בדואר ("טיפול מיוחד"). לקוח רגיל חייב לשלם לפני ששולחים לו את הספרים ("טיפול רגיל"). לקוח נחשב כקבוע אם הוא מקיים את התנאים הבאים:

- קונה סחורה שערכה עולה על 8,000 ש"ח לשנה
- בעבר שילם את כל החשבונות בזמן (תוך 30 יום)
- בעל וותק של יותר מ-5 שנים.

חברת ההפצה מוכנה לתת טיפול מיוחד גם ללקוחות שמקיימים את

השילובים הבאים:

- (1) ו-(2)
- (1) ו-(3)
- (2) ו-(3)
- (2).

כתוב תוכנית שתקבל כקלט:

- ערך הסחורה שהלקוח קונה במשך השנה.
- האם בעבר הלקוח שילם את החשבונות בזמן? (כן או לא)
- מספר השנים שהלקוח קונה ספרים מן החברה.

כתוב פונקציה שמוציאה כפלט את המשפט הבא:

תן ללקוח טיפול מיוחד: XXXX (True או False)



רמז:

לפני כתיבת התוכנית, צמצם את מספר התנאים ל-2.

תאריכים

משתנים יכולים להיות מסוג Date המחזיק תאריכים. האופרטורים "+" ו "-" גורמים לחיבור וחיסור ימים מתאריך כלשהו. ניתן גם להפחית תאריך אחד מתאריך שני בעזרת האופרטור "-" כאשר שני האופרנדים הם מסוג Date. הסוג Date מחזיק גם זמן במבנה hh.mm.ss PM/ AM, ואפשר לחבר זמנים אחד לשני, או להחסיר זמנים אחד מן השני, כאשר האופרנד הראשון נחשב כשעה בשעון, והשני נחשב כמספר של שעות, דקות ושניות להוספה או להורדה מהאופרנד הראשון.

תוכנית A10: קליטת משתני שיחות.

```
Public Function A10(datCall As Date, datLength As Date)
' Calculates the length of a phone call. Assume 24 hour clock.
Dim datEnd As Date, DatTime As Date

datEnd = datCall + datLength

Debug.Print " שעת סיום השיחה: "; datEnd

DatTime = #1/31/97 1:02:45 PM#

Debug.Print " תאריך ושעת סיום השיחה: "; DatTime

End Function
```

הסבר: תוכנית זו קולטת שני זמנים, האחד - זמן התחלת שיחה, והשני - אורך השיחה, כפי שעשינו בתוכנית A04. במקום תוכנית ארוכה, החלק האופרטיבי של תוכנית זו תופס שורה אחת בלבד. מריצים את התוכנית על ידי הזנת שני הזמנים כקבועים, כדלהלן: (#0:12:55# ,#12:23:23#) A10, כאשר ערך הפרמטר הראשון מציין את שעת התחלת השיחה, והשני מציין את אורך השיחה. שים לב שקבוע מסוג Date מחייב סולמיות משני צידיו. מחברים את שני הקלטים, ומקבלים את זמן סיום השיחה. החלק השני של התוכנית מציג איך להגדיר קבוע מסוג Date שמחזיק תאריך עם זמן.

תוכנית A11: שימוש בפונקציות שונות לביצוע מניפולציות על תאריכים.

```
Public Function A11()
Dim Ndate As Date, Ndate2 As Date
Dim NewDate As Date, NewDate2 As Date, EndWork As Date
Dim intYears As Integer, intMonths As Integer, intDays As Integer, Num As Integer
Dim Yom As Integer, Chodesh As Integer, Shana As Integer, YomBashavua As Integer
Dim Century As Integer
Dim Tarich As String
' Finding present time and date
A: MsgBox "התאריך הוא: " & Date$ & Chr$(13) & " השעה היא: " & Time$
B: MsgBox " השעה והתאריך הם: " & Now
' Simple arithmetic can be used for adding and subtracting days.
C: Ndate = Date + 25
D: MsgBox "New Date is: " & Ndate
E: Ndate = Date - 25
F: MsgBox "New Date is: " & Ndate

' 2 dates may be subtracted to determine the difference of days
G: intDays = Date - Ndate
H: MsgBox "The number of days difference is: " & intDays
' Adding and subtracting months or years

' To add  months and years use DateAdd
' To add dates use DateAdd:d for days, m for months, yyyy for years

' not necessary for days - but possible
I: Ndate2 = DateAdd("d", 25, Date) 'adds 25 days ("d") to present date
J: MsgBox "New Date is: " & Ndate2
' Adding years
   EndWork = DateAdd("yyyy", 10, Date) 'adds 10 years ("yyyy") to present date
K:  MsgBox "End work: " & EndWork
' To determine the difference in months or years between 2 dates, ' use the DateDiff
' function  Number of years between 2 dates
L: intYears = DateDiff("yyyy", Ndate, EndWork)
```

```
' Number of months between 2 dates
M: intMonths = DateDiff("m", Ndate, EndWork)

' Number of days - not necessary, but it works
N: intDays = DateDiff("d", Ndate, EndWork)
O: MsgBox "Elapsed time in years: " & intYears
P: MsgBox "Elapsed time in months: " & intMonths
Q: MsgBox "Elapsed time in days: " & intDays
' The year difference function measures in terms of the difference in calendar years, i.e.
' 1996 - 1995 gives 1
' independent of the months
R: NewDate2 = Date + 355  'adds 355 days ("d") to present date
S: intYears = DateDiff("yyyy", Date, NewDate2)
t: MsgBox "Elapsed time in years: " & intYears

' The Datepart function extracts day of month (d),
' day of year (y), year (yyyy), quarter (q), weekday (w), month (m)
' week (ww), hour (h), minute (n),
' second (s), depending on the parameter used
' Instead of datepart with parameter d can use Day function
' Instead of datepart with parameter w can use WeekDay function
' Instead of datepart with parameter m can use Month function
' Instead of datepart with parameter yyyy can use Year function

U: Num = DatePart("w", Now) ' 1 is Sunday, 2 is Monday, 3 is Tuesday, etc.
V: MsgBox "The day of the week today is: " & Num
W: Num = DatePart("d", Now)
X: MsgBox "The day of the month today is: " & Num
' DateSerial changes numbers into a date.
Y: NewDate = DateSerial(1990 + 7, 12 - 2, 26 + 11)
Z: MsgBox "The desired date is: " & NewDate
AA: Yom = Day(NewDate)
BB: Chodesh = Month(NewDate)
CC: Shana = Year(NewDate)
DD: YomBashavua = WeekDay(NewDate)
EE: MsgBox "The day is: " & Yom & Chr$(13) & "The month is: " & Chodesh
FF: MsgBox "The year is: " & Shana & Chr$(13) & "The day of the week is: " &
YomBashavua
' The DateValue function changes a string value to a date value
GG: Tarich = "17/8/44"
HH: NewDate = DateValue(Tarich)
II: MsgBox "The desired date is: " & NewDate
' Of course, the best way to define a date is by using the date literal
JJ: NewDate = #12/31/70#
```

```
KK: MsgBox "The desired date is: " & NewDate
' isdate() determines if its argument is a data type or can be converted to such
LL: NewDate = InputBox("Enter a date", "Century Calculation", Date)
MM: Century = ((Year(NewDate) - 1) \ 100) + 1
MsgBox Century, 65, "Desired Century"

End Function
```

הסבר:

- **A**: הפונקציה Date מחזירה את התאריך הנוכחי, הפונקציה Time מחזירה את השעה הנוכחית. התו הקשור לערך 13 ב-ASCII גורם לפלט לעבור לשורה הבאה.
- **B**: הפונקציה Now מחזירה את התאריך וגם את השעה הנוכחית.
- **D-C**: חיבור ימים לתאריך.
- **F-E**: חיסור ימים מהתאריך.
- **H-G**: הפרש הימים בין שני התאריכים.
- **K-I**: בעזרת הפונקציה DateAdd מוסיפים ימים, חודשים, או שנים.
- **T-L**: הפרש בימים, חודשים, או שנים בין שני תאריכים בעזרת הפונקציה DateDiff.
- **X-U**: פונקצית DatePart מוציאה מידע מסוגים שונים מנתון מסוג Date.
- **Z-Y**: הפונקציה DateSerial מאפשרת בניית ערך מסוג Date על בסיס מספרים שלמים עבור השנה, החודש והיום. הפונקציה מבצעת גם חישובים, ואם מספר הימים או החודשים גדול מדי, היא מעבירה את זה הלאה ליחידה הבאה, דהיינו לחודשים ו/או שנים, בהתאמה.
- **AA-FF**: דרכים מקבילות להוצאת מידע מנתון מסוג Date.
- **GG-II**: העברת מחרוזת בצורת תאריך למשתנה מסוג Date.
- **JJ-KK**: טעינת משתנה מסוג Date בערך קבוע.
- **LL-MM**: קליטת תאריך, וקביעת המאה של אותו תאריך. כזכור, הפונקציה Date מחזירה את התאריך הנוכחי כברירת המחדל עבור תיבת הקלט.



תרגיל 22:

בתרגול הבא, קלוט כל תאריך מחוץ לשיגרה.

- כתוב שיגרה המציגה את היום בשבוע בו נולדת?
- כתוב שיגרה שמציגה בכמה שנים אביך מבוגר ממך (X).
- כתוב שיגרה שמציגה בכמה חודשים אביך מבוגר ממך (Y).

- כתוב שיגרה שמציגה בכמה ימים אביך מבוגר ממך (Z).
- היום ועוד X שנים, ייפול באיזה תאריך?
- היום ועוד Y חודשים, ייפול באיזה תאריך?
- היום ועוד Z ימים, ייפול באיזה תאריך?
- היום ועוד X שנים, Y חודשים ו-Z ימים, ייפול באיזה תאריך?

משתנים מסוג Variant

משתנה מסוג Variant יכול להחזיק משתנה מכל אחד מהסוגים שהוזכרו קודם, כלומר, נומריים, מחרוזות, תאריכים, ובוליאניים. בנוסף, גם אם משתנה מסוג Variant תורגם לסוג מסוים, אפשר לשנות את סוגו כתוצאה מדרישות התוכנה.

תוכנית A12: שימוש בסוג Variant.

```
Public Function A12()
Dim varAnyType
' equivalent to Dim varAnyType as Variant, which is the default

' conversion from double to string
A: varAnyType = 21.354
varAnyType = varAnyType & " is the root of 456"
Debug.Print varAnyType

' concatenation of the string variables
B: varAnyType = "18.89"
varAnyType = varAnyType + "34"
Debug.Print varAnyType

' conversion of the string variable to numeric type
C: varAnyType = "18.89"
varAnyType = varAnyType + 34
Debug.Print varAnyType

' conversion of the string to numeric type
D: varAnyType = 18.89
varAnyType = varAnyType + "34"
Debug.Print varAnyType


' conversion of the numeric variable to string type
E: varAnyType = 18.89
varAnyType = varAnyType & "34"
Debug.Print varAnyType

' determines integer type (code 2)
```

```
F:  varAnyType = 34 + 45
Debug.Print VarType(varAnyType)

' determines long integer type (code 3)
G: varAnyType = 39777
Debug.Print VarType(varAnyType)

' determines double type (code 5)
H: varAnyType = 397.567
Debug.Print VarType(varAnyType)

End Function
```

הסבר: שים לב שלא הגדרנו את סוג המשתנה, משום ש-Variant הוא ברירת המחדל. לכן, בפקודה כמו: Dim A, B, C as Integer המשמעות אינה ש: A, B, ו-C הם מסוג Integer, אלא ש-A ו-B הם מסוג Variant ורק C הוא מסוג שלם.

- **A**: לאחר ההצבה הראשונה, סוג המשתנה הוא מספרי, אך כתוצאה מהקישור בשורה השנייה, הסוג הופך למחרוזת.
- **B**: לאחר ההצבה הראשונה, סוג המשתנה הוא מחרוזת, ובפקודת ההשמה השנייה מתבצע קישור מחרוזות המספק את התשובה 18.8934.
- **C**: לאחר ההצבה הראשונה, סוג המשתנה הוא מחרוזת, ובפקודת ההשמה השנייה, התוכנה מבינה שהיא צריכה לשנות את המשתנה לסוג מספרי כדי לחבר את שני המספרים ולקבל 52.89.
- **D**: לאחר ההצבה הראשונה, סוג המשתנה הוא מספרי, ובפקודת ההשמה השנייה, התוכנה מבינה שהיא צריכה לשנות את המחרוזת "34" לערך מספרי, כדי לחבר את שני המספרים ולקבל 52.89.
- **E**: לאחר ההצבה הראשונה, סוג המשתנה הוא מספרי, ובפקודת ההשמה השנייה, התוכנה מבינה שהיא צריכה לשנות את סוג המשתנה למחרוזת, כדי להפעיל את האופרטור & ולקבל 18.8934. שים לב שההבדל היחיד בין D ו-E הוא שהשתמשנו באופרטור מחרוזות & ב-E, כאשר ב-D השתמשנו באופרטור + של מספרים.
- **H-F**: סוגים שונים שהמשתנה מסוג Variant מקבל. הפונקציה VarType מיועדת להודיע על הסוג הנוכחי של משתנה שהוגדר כ-Variant.

סיכום:

- Access ממירה משתנה מסוג Variant לסוג המתאים, בביטויים אריתמטיים ובקישורים (Concatenations).
- & יגרום לקישור מחרוזות, ויהפוך ערך מספרי למחרוזת כדי לבצע את המטלה.
- + יגרום לחיבור אם שני הערכים מספריים, או אם אחד מספרי והשני מחרוזת שמכילה מספרים. אם שניהם מחרוזות, הוא יבצע קישור (Concatenation). אם אחד מספרי והשני מחרוזת שלא ניתן להפוך למספר, תיגרם שגיאה.

איפוס משתנים

יש 4 סוגי משתנים מאופסים, והם:

- 0 - ערך מספרי. ברירת מחדל עבור משתנים מסוגים מספריים כולל סוג מטבע.
- מחרוזות באורך אפס (Zero Length Strings) - ברירת מחדל למחרוזות. מסומן ב- "" בפקודת השמה.
- ריק (Empty) - ברירת המחדל עבור משתנים מסוג Variant. הערך משתנה ל-0 או למחרוזת באורך אפס, אם הוא מהוה חלק מביטוי, לפי סוג הביטוי.
- Null - ערך השדות במסד הנתונים שלא הוזנו בהם נתונים. רק משתנה מסוג Variant יכול לקבל ערך זה, בעזרת פקודת השמה. הפעולה מתבצעת באמצעות הצבת המילה השמורה Null, או ביטוי שערכו Null, בצד ימין של פקודת ההשמה.

תוכנית A13: סוגי איפוס שונים.

```
Public Function A13()
Dim varAnyType, varAnyType2, varAnyType3 ' default type is Variant

' indicating that varAnyType is empty in 2 ways (varType function returns value of 0)
A: Debug.Print IsEmpty(varAnyType)
B: Debug.Print VarType(varAnyType), VarType(varAnyType) = vbEmpty

' variant type assumes 0 default value
C: varAnyType = varAnyType + 21.354
D: Debug.Print varAnyType, VarType(varAnyType), VarType(varAnyType) = vbDouble

' setting the variant to a zero-length string, or to 0,  does not make it empty again
E: varAnyType =""
F: Debug.Print IsEmpty(varAnyType), VarType(varAnyType)

' Only assigning the variant to another variant makes it empty
G: varAnyType = varAnyType2
H: Debug.Print IsEmpty(varAnyType)

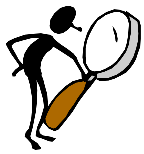
' any null operand makes the result null with arithmetic operators
I: varAnyType = Null
J: varAnyType = varAnyType + 21.354
K: Debug.Print varAnyType
' IsNull function for detecting nulls, or vartype which returns 1
L: Debug.Print IsNull(varAnyType), VarType(varAnyType), VarType(varAnyType) _
 = vbNull
```

```
' Null operand with string operator (&) _
  accepted as zero-length string
M: varAnyType2 = "Yankel"
N: varAnyType3 = varAnyType & varAnyType2
O: Debug.Print varAnyType3, VarType(varAnyType3)

End Function
```

הסבר:

- **B-A**: הערך ההתחלתי של משתנה מסוג Variant הוא ריק. הפונקציה IsEmpty תחזיר את הערך **נכון** (True), והפונקציה VarType מחזירה ערך 0, שהוא קוד הסוג **ריק** – Empty. הקבוע vbEmpty מוגדר עבור הפונקציה VarType וערכו 0, ולכן הביטוי האחרון יחזיר ערך **נכון** (True).
- **D-C**: כאשר מחברים מספר ממשי למשתנה ריק מסוג Variant, המשתנה מסוג Variant מקבל כברירת מחדל ערך ממשי 0. הפונקציה VarType מחזירה כעת ערך 5 (המכונה vbDouble) המסמל משתנה ממשי.
- **F-E**: גם אם משווים משתנה Variant למחרוזת ריקה, הוא לא חוזר להיות ריק. במקרה זה, הוא נשאר להיות מסוג מחרוזת (VarType של 8, המכונה vbString).
- **H-G**: הדרך היחידה להחזיר משתנה Variant להיות ריק היא על ידי השוואתו למשתנה Variant אחר שעדיין ריק, כלומר, לא קיבל אף פעם ערך כלשהו.
- **K-I**: כשלמשתנה Variant ערך Null, תוצאת החיבור למספר ממשי תהיה Null.
- **L**: הפונקציה IsNull מגלה אם משתנה מסוג Variant שווה ל-Null. הפונקציה VarType תחזיר ערך 1 (vbNull) אם ערך המשתנה מסוג Variant שווה ל-Null .
- **M-O**: משתנה מסוג Variant שערכו Null יכול להיות מקושר למשתנה מסוג מחרוזת, אם האופרטור המקשר הוא &, והמשתנה נחשב מחרוזת באורך 0. VarType מחזירה ערך 8 (מחרוזות).



הערה:

קבועים נוספים עבור הפונקציה VarType הם: vbCurrency, vbDate, ו-vbBoolean. כדי לראות את כל הקבועים של פונקציה זו, עיין בתפריט **עזרה**, **אינדקס**, תחת הפונקציה **VarType**.

פונקציות שמספקות תוצאה מסוג Variant

רוב הפונקציות עבור מחרוזות שראינו, יכולות להחזיר ערך מסוג Variant. ההבדל היחיד בין פונקציה שמחזירה מחרוזת ופונקציה שמחזירה ערך מסוג Variant, הוא שכותבים את הראשונה עם סימן $ בסוף השם. היות ומשתנה מסוג Variant יודע לשנות את סוגו בעת הצורך, מומלץ להשתמש בו בעבודה עם משוואות. לפונקציות הבאות יש גירסה למחרוזות עם סימן $ בסוף, וגירסה בלי סימן $:

Chr$	Lcase$	Mid$	Space$	Time$
Date$	Left$	Right$	Sring$	Trim$
Input$	Ltrim$	Rtrim$	Str$	Ucase$

יתרונות וחסרונות של משתנה מסוג Variant

יתרונות:

- אין צורך להתייחס לסוג המשתנה.
- משתנה זה הוא הסוג היחיד שמסוגל לקבל על עצמו ערך Null. היות ושדות בבסיסי נתונים יכולים להיות Null, ניתן לבדוק את ערכם רק באמצעותו.

חסרונות:

- הוא מאט את קצב הביצוע, משום שעל Access לקבוע בעצמה את סוג המשתנה.
- מאפשר למפתח לבצע שגיאות שהיו מתגלות לו סוג המשתנה היה נקבע במדויק. Access תקבע סוג אפשרי לפי הקוד שכתבנו, ולא תמיד הוא יהיה הסוג שרצינו.

קבועים (Constants)

מותר כמובן להשתמש בקבוע מבלי להצהיר עליו מראש. אך מומלץ להשתמש בהצהרת Const, באופן דומה להצהרת Dim על משתנים. כאשר מצהירים על הקבוע מראש, מונעים משם זה לקבל כל ערך, חוץ מהערך הנקוב, ובזה שונה הצהרת Const מהצהרת Dim. ישנם שלושה סוגי קבועים:

- **קבועים סמליים הבנויים בעזרת פקודת Const**: בונים קבוע סמלי בשל אחת מהסיבות כדלהלן:
  - כאשר משתמשים בקבוע פעמים רבות במשך התוכנית (במיוחד, אם הוא מספר ארוך, למשל π=3.1415926536), קל יותר להגדיר אותו פעם אחת בתחילת התוכנית, ולהימנע מכתיבה חוזרת ומטעויות אפשריות.
  - אם משתמשים בקבוע פעמים רבות בתוכנית, והוא עשוי להשתנות מדי פעם בין הרצה להרצה. אפשר לשנות רק את הצהרת Const ולא את כל האזכורים של אותו קבוע בתוך התוכנית.

- כדי למנוע כפל משמעויות לשם, כפי שמקובל בתפיסת התכנות המבני. הגדרת שם כ-Const מונעת ממנו מלשמש למטרה אחרת.
- אם יש חשש שמשמעות הקבוע לא תובן, השם שניתן לו עשוי לעזור. למשל, השם MATANA בפקודת ההשמה שלהלן מסביר הרבה יותר מהמספר 25 בגירסה השנייה של הפקודה:

```
1.    Maskoret = Shiur * Shaot + Matana
2.    Maskoret = Shiur * Shaot + 25
```

- **קבועים פנימיים ל-Access**: קבועים המוגדרים על ידי המהדר של Access. למשל, כפי שראינו בפונקציית VarType, לכל תוצאה אפשרית יש שם. גם עבור הפעולה MsgBox ראינו שקיימים קבועים פנימיים.
- **קבועים המוגדרים על ידי מערכת ההפעלה**: למשל True, False ו-Null.

תוכנית A14: הגדרת csPI כקבוע.

```
Public Function A14(dblRadius As Double)
Dim dblArea As Double, dblVolume As Double
Const csPI = 3.1415926536
dblArea = csPI * dblRadius ^ 2
dblVolume = csPI * dblRadius ^ 3 * 4 / 3
Debug.Print " Area is: "; Format(dblArea, "##.##"); _
        "Volume is: "; Format$(dblVolume, "##.##")

End Function
```

הסבר: הקידומת cs מסמנת שמדובר בקונסטנטה (קבוע). הפונקציה Format מאפשרת להדפיס את התשובה עם שתי ספרות עשרוניות בלבד. המחרוזת "##.##" נקראת מחרוזת פורמט, ואומרת שלפני ואחרי הנקודה עשרונית תופענה שתי ספרות. אם יש יותר משתי ספרות לפני, מאפשרים גלישות, שהרי אף אחד אינו רוצה לקצץ מספרים משמעותיים מהתוצאה. יש הרבה סמלים שאפשר להשתמש בהם במחרוזת הפורמט (עיין ב**עזרה**, **אינדקס**, **Format**, ואז בחר מתוך **See Also** את הכניסות הבאות: **User Defined Numeric Formats**, **User Defined String Formats**, **User Defined Date/Time Formats**).



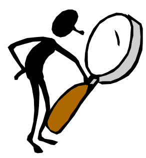
תרגיל 23:

עבור על התרגילים 2-4, 10 ו-14 והשתמש בקבועים לפי הצורך. דאג גם שכל תוצאה תהיה בעלת 2 ספרות עשרוניות בלבד בעזרת פקודת פורמט.

פרק 2

מבני פיקוח

מבני פיקוח (Control Structures) מכילים פקודות שמשפיעות על כיוון זרימת התוכנית. הפקודות שהשתמשנו בהן עד עכשיו התבצעו בצורה סדרתית, אחת אחרי השנייה. פקודות פיקוח מאפשרות שינוי בכיוון זרימת התוכנית, ומאפשרות הסתעפויות ולולאות. מבני פיקוח הם משני סוגים: הסתעפויות מותנות (מבנה IF ו-SELECT) ולולאות (DO..WHILE ו-For). יש גם פקודה להסתעפות מוחלטת (GoTo) שהשימוש בה אינו מומלץ.



הערה:
בפקודות הבאות אפשר לכתוב יותר מפקודה אחת באותה שורה אם מפרידים בין הפקודות באמצעות נקודתיים (:).

הסתעפות מותנית

פקודת IF

מבנה בסיסי:

```
IF  ביטוי בוליאני  Then
  פקודות
Else
  פקודות
End  If
```

תוכנית B01: דוגמה לשימוש ב-IF.

```
Public Function B01()
 Dim QtySpecial As Integer, Bonus As Currency
  QtySpecial = InputBox("Enter quantity of special items sold", "Special Sales", "0")
 If QtySpecial Then  ' Equivalent to saying QtySpecial <> 0, as in C
     Bonus = 500
  Else
     Bonus = 0
End If
' The message function takes only string or variant
' Format changes its input to variant, Format$ to string
 MsgBox Format$(Bonus, "$#.00"), 65, "Calculated Bonus"
End Function
```

בתוכנית B01 נקלוט את כמות המכירות המיוחדות שביצע סוכן. אם הסוכן ביצע מכירות כאלו, מגיע לו בונוס של $500, אחרת, לא מגיע לו בונוס. הפונקציה Format הופכת את הבונוס למספר עם סימן $ בצד שמאל. הסימן # מאפשר לקצץ אפסים מובילים, כלומר אם יש רק ספרה אחת, רק מקום אחד בצד שמאל של הנקודה העשרונית יתמלא, ואם יש 3 ספרות, כמו פה, 3 מקומות ייתפסו. הספרה 0 מימין לנקודה העשרונית מבטיחה שספרה כלשהי תופיע במקום זה, כך שבמקרה זה שתי ספרות תופענה בצד ימין של הנקודה העשרונית.

מבנה IF מורכב:

```
IF   ביטוי בוליאני  Then
  פקודות
[ElseIf   ביטוי בוליאני Then
   פקודות]. . .
]Else
   פקודות]
End  If
```

הסבר: הביטוי ElseIf מופעל אם הביטוי הבוליאני אינו שווה ל-True. שלוש הנקודות (. . . שנקראות Ellipsis) מציינות שמותר לחזור על הביטוי לפי הצורך.

תוכנית B02: דוגמה לשימוש ב-ElseIf.

```
Public Function B02()
' A book is due on the first day of the next month, unless it is
' Saturday or Sunday
 Dim AnyDate As Date, DueDate As Date
 AnyDate = InputBox$("Enter a date", "Date Due Calculation", Date)
 DueDate = DateSerial(Year(AnyDate), Month(AnyDate) + 1, 1)
 If WeekDay(DueDate) = 1 Then
```

```
    DueDate = DueDate + 1
 ElseIf WeekDay(DueDate) = 7 Then
    DueDate = DueDate + 2
 End If
 MsgBox DueDate, 65, "Calculated Due Date"

End Function
```

בתוכנית B02 אנו קולטים את התאריך בעזרת InputBox. הפונקציה DateSerial מסוגלת לקחת שלושה מספרים שלמים, אחד עבור שנה, אחד עבור חודש, ואחד עבור יום, ולהפוך אותם לפורמט תאריך תקין. אם מספר החודשים גדול מ-12, או מספר הימים גדול מ-31, פונקציה זו יודעת להוסיף יותר משנה או מחודש, בהתאמה. הפונקציות Year ו-Month מסוגלות להוציא את השנה והחודש כמספרים שלמים מתוך תאריכים תקינים. תאריך החזרת הספר הוא בראשון לחודש, ולכן אנו מכניסים 1 כפרמטר השלישי בפונקציה DateSerial. אך אין אפשרות להחזיר ספר בשישי או בשבת, ולכן, אם הראשון לחודש הוא ביום שישי, מוסיפים 2 לתאריך ההחזרה, ואם הוא בשבת, מוסיפים אחד.

תוכנית B03: דוגמה נוספת לשימוש במבנה If מורכב.

```
Public Function B03()
Rem 6 is the code for pressing "Yes", while 35 builds a box with
Rem "Yes" in it
 If (MsgBox("האם אתה מוכן?", vbQuestion + vbYesNoCancel) = vbYes) Then
    MsgBox "לחצת על כן"
 Else
    MsgBox "לחצת על לא"
End If
End Function
```

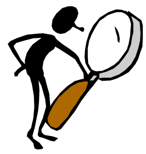
אנחנו משתמשים ב-MsgBox בצורה חדשה, לא כפקודה אלא כפונקציה שמחזירה ערך. הערכים שהפונקציה מחזירה תלויים בלחצן שעליו לחץ המשתמש בתיבה המוצגת, לפי הקודים דלהלן:

1. נלחץ **אישור**
2. נלחץ **ביטול**
3. נלחץ **בטל**
4. נלחץ **נסה שנית**
5. נלחץ **התעלם**
6. נלחץ **כן**
7. נלחץ **לא**.

בפונקציה זו השתמשנו בקבועים פנימיים של Access. אנו משתמשים ב-MsgBox עם קוד 35, זאת אומרת עם סמל אזהרה (סימן שאלה, עם קבוע פנימי vbQuestion), ועם הלחצנים **כן**, **לא**, ו**בטל** (קבוע פנימי vbYesNoCancel). בפקודת IF אנו שואלים את המשתמש אם הוא מוכן, לאחר תגובתו אנו מודיעים עליה. תשובת **כן** גורמת לפונקציה להחזיר ערך 6 (vbYes), כשכל תשובה אחרת מתורגמת כאות שהמשתמש אינו מוכן.

מבנה בסיסי **(חייב להופיע בשורה אחת):**

```
If  ביטוי בוליאני  Then  פקודות  [Else  פקודות ]
```



הערה:

במבנה זה, כשהפקודות ב-Then ו-Else מופרדות על ידי נקודתיים, אך כל המבנה מופיע בשורה אחת (אפילו אם השורה ממשיכה לשורה הבאה על ידי שימוש בסימן _), אין צורך ב-EndIf.



תרגיל 1:

בתרגיל זה, קלוט תאריכי קלט מחוץ לשיגרה.

- כתוב שיגרה שמציגה בכמה שנים (X), חודשים (Y), וימים (Z) אביך מבוגר ממך.
- היום ועוד X שנים, Y חודשים ו-Z ימים ייפול באיזה תאריך?

תרגיל 2:

כתוב תוכנית שמקבלת כקלט שני מספרים שלמים A ו-B, ומוציאה כפלט משתנה בוליאני שערכו TRUE, אם B מתחלק ב-A ללא שארית.

הערה: ניתן לכתוב שתי גרסאות כמענה לשאלה זו, אחת ללא שימוש בפקודת IF, ואחת עם.

תרגיל 3:

כתוב תוכנית לקריאת קבוצות של 3 מספרים המייצגים את אורכי צלעות המשולש. הפלט לכל קבוצה הוא אחת מההודעות הבאות:

- הצלעות מהוות משולש שווה צלעות.
- הצלעות מהוות משולש שווה שוקיים.
- הצלעות מהוות משולש שונה צלעות.
- הצלעות אינן יכולות להרכיב משולש (זאת אומרת סכום האורכים של זוג כלשהו, קטן מאורך הצלע השלישית).



תרגיל 4:

כתוב שיגרה שקוראת זוג מילים באנגלית ומדפיסה אותן זו ליד זו לפי סדר אלפביתי, משמאל לימין. למשל, שתי המילים Very Good תודפסנה בסדר הפוך, אך שתי מילים אלו: Big Dog תישארנה בסדר זה. דאג שבפלט יהיו 5 רווחים בין המילה הראשונה לשנייה.

תרגיל 5:

נתונים שלושה מספרים: A, B, ו-C. כתוב שיגרה המדפיסה אותם בסדר עולה.

לולאות

DO

מבנה ראשון:

```
Do [{While|Until} ביטוי בוליאני]
  [פקודות]
  [Exit Do]
  [פקודות]
Loop
```

תוכנית B04: שימוש במבנה ראשון של Do.

```
Public Function B04()
Dim Count As Integer
Count = 1
Do While (Count <= 10)
' Debug.Print "ספירה במספר:"; Count  'Allows viewing from Immediate box
                ' button, or VIEW/IMMEDIATE
MsgBox "Counting at: " & Count, 1, "Do While Statement"
Count = Count + 1
Loop

End Function
```

פונקציה זו מוסיפה 1 ל-Count בכל מעבר בלולאה, ומדפיסה את הערך המעודכן של Count, כך שהיא מדפיסה את המספרים מ-1 עד 10. הפלט נראה כך:

ספירה במספר: 1

ספירה במספר: 2

ספירה במספר: 3

ספירה במספר: 4

ספירה במספר: 5

ספירה במספר: 6

ספירה במספר: 7

ספירה במספר: 8

ספירה במספר: 9

ספירה במספר: 10

דוגמאות נוספות:

תוכנית B05: איתור הגורם המשותף הגדול ביותר של שני מספרים.

```
Public Function B05()
Dim A As Integer, B As Integer
A = InputBox("הזן מספר שלם, בבקשה")
B = InputBox("הזן עוד מספר שלם, בבקשה")
Do While A <> B
  Do While A > B
    A = A - B
   Loop
 Do While B > A
    B = B - A
   Loop
Loop
MsgBox "הגורם המשותף הגדול ביותר שווה ל: " & A
End Function
```

נניח שיש למצוא את הגורם המשותף הגדול ביותר של שני מספרים: GCF (Greatest Common Factor). כדי לעשות זאת, נתבסס על האלגוריתם של אוקלידוס שאומר:

- אם X הוא GCF של A ו-B, אז ההפרש בין שני המספרים מתחלק ב-X.
- GCF של שני מספרים זהים שווה לאותו מספר עצמו.

ההוכחה לכלל הראשון נובעת מהעובדה, שאם X הוא GCF של A ו-B זה מחייב שיהיו שני מספרים A1 ו-B1, כך ש: A=XA1 ו-B=XB1 ומכאן נקבל: B-A=X(B1-A1). המשמעות היא, שההפרש B-A מתחלק ב-X. הכלל השני מובן מאליו. השיטה המוצעת לחישוב GCF היא תהליך חוזר של חישוב ההפרש בין שני המספרים הנתונים, והחלפת הגדול שבהם על ידי ההפרש, עד ששניהם שווים. לדוגמה, נתונים המספרים A ו-B. אם A>B, נחליף את A בהפרש A-B. נשווה את B לערך החדש של A, ונמשיך

להחליף את הערך הגדול בהפרש שבין המספר הגדול למספר הקטן, עד שנגיע למצב שבו שני הערכים שווים. כעת קבלנו את GCF של זוג המספרים המקוריים A ו-B.

תוכנית B06: שינוי צורת השם.

```
Public Function B06()
' takes an entire name like John Will Friedman and outputs Friedman, J W

Dim strName As String
Dim FirstName As String
Dim MidName As String
Dim LastName As String
Dim Char As String
Dim Length As Integer
Dim Location As Integer

1: Location = 1
2: strName = InputBox ("בבקשה להזין שם פרטי, אמצעי ומשפחה", "סידור השם", "")
3: strName = Trim(strName)
4: Length = Len(strName)
5: FirstName = Left$(strName, 1)
6: Location = InStr(strName, " ")
7: Char =" "
8: Do Until Char <> " "   ' maybe there are a number of blanks
     Location = Location + 1
     Char = Mid(strName, Location, 1)
   Loop
9: MidName = Char
10: Location = InStr(Location + 1, strName, " ")
11: Char =" "
12: Do Until Char <>" "
      Location = Location + 1
      Char = Mid(strName, Location, 1)
    Loop
13: LastName = Right$(strName, Length - Location + 1)

14: MsgBox LastName & ", " & FirstName & " " & MidName
End Function
```

תוכנית B06 קולטת את השם חיים יעקב כהן, ופולטת כ: כהן, ח י. הסבר לפי שורות:

- 2: הזנת שם פרטי, אמצעי ומשפחה.
- 3: הורדת רווחים משני הצדדים.
- 4: קביעת אורך המחרוזת החדשה, לאחר הורדת הרווחים.
- 5: האות הראשונה בשם הראשון היא התו השמאלי ביותר במחרוזת החדשה.

- 6: מחפשים את מיקום הרווח הראשון.
- 7-8: היות ויש אפשרות שיש יותר מרווח אחד בין השם הראשון והשני, ממשיכים להתקדם עד שמגיעים לתו ששונה מרווח.
- 9: התו ששונה מרווח הוא התו הראשון בשם האמצעי.
- 10: מחפשים את הרווח הראשון לאחר השם האמצעי.
- 11-12: כמו 7-8.
- 13: שם המשפחה הוא החלק הימני של המחרוזת, החל מהתו הראשון ששונה מרווח לאחר השם האמצעי.

תוכנית B07: קביעת מספר המילים במשפט.

```
Public Function B07()
' takes a sentence and determines how many words are in it
Dim strSentence As String
Dim FirstName As String
Dim MidName As String
Dim LastName As String
Dim Char As String
Dim NumWords As Integer
Dim Location As Integer
1: Location = 1
2: strSentence = InputBox("בבקשה להזין משפט שלם", "מספר מילים", "")
3: strSentence = Trim(strSentence) +" "
4: Do Until Location = 0
5:     Location = InStr(Location, strSentence," " )
6:     NumWords = NumWords + 1
7:     Char =" "
8:     Do Until Char <> " " Or Location > Len(strSentence)
9:         Location = Location + 1
10:        Char = Mid(strSentence, Location, 1)
       Loop
11:    If Location > Len(strSentence) Then Location = 0 'needed at end
Loop
12: MsgBox " מספר המילים במשפט הוא:" & NumWords
End Function
```

- 2: קליטת המשפט.
- 3: הורדת כל הרווחים בהתחלה ובסוף, ואז הוספת רווח אחד בסוף.
- 4: רווח מסמן סוף מילה. המשתנה Location יהיה 0 אם לא מצאנו מה שחיפשנו, במקרה שלנו, רווח. לכן, נמשיך את הלולאה עד ש: Location שווה 0.
- 5: מחפשים את הרווח הראשון.

- 6 : מוסיפים 1 למספר המילים.
- 7-10 : לוקחים בחשבון את האפשרות שיש יותר מרווח אחד בין מילה למילה.
- 11 : אם גמרנו את המשפט, מוציאה אותנו מהלולאה החיצונית.

צורה שנייה - לפחות ביצוע אחד

```
Do
  [פקודות]
  [Exit Do]
  [פקודות]
Loop  [{While|Until} ביטוי בוליאני]
```

תוכנית B08:

```
Function 08()
Dim Count As Integer
Count = 1
Do
   Debug.Print Count; "ספירה במספר"
'   MsgBox "ספירה במספר  " & Count, 1, "Do Until Statement"
   Count = Count + 1
Loop Until Count > 10
End Function
```

זהה לדוגמה הקודמת, אך הבדיקה מתבצעת בסוף, ולכן גם אם נתחיל בערך התחלתי 11 עבור Count, הפונקציה תרוץ לפחות פעם אחת, ותודפס תוצאה 11. בדוגמה הקודמת אם נתחיל בערך התחלתי 11 עבור Count, לא תודפס תוצאה כלשהי.

צורה שלישית, פשוטה יותר אך ללא אפשרות של יציאת חירום

```
While...Wend

While ביטוי בוליאני
פקודות
Wend
```

פקודה בה הערך ההתחלתי והסופי מוגדרים כחלק מהמבנה עצמו

```
For..Next
For counter = first TO last [Step amount]
[פקודות]
[Exit For]
[פקודות]
Next [counter]
```

הסבר: המונה counter מתחיל מהערך first, ועולה בצעדים של step amount עד שהוא מגיע לערך last, אז נפסקת הלולאה.

תוכנית B09:

```
Function B09()
Dim Count As Integer, Ans As String
For Count = 1 To 10
 'Debug.Print "Counting at"; count
  Beep
  MsgBox "ספירה ב: " & Count, 1, "For Next Statement"
  Ans = InputBox("האם אתה רוצה לראות עוד אחד (כ/ל)?")
  If Ans = "ל" Then Exit For 'Chance to leave early. 1 line If
Next Count
End Function
```

עוד פקודה שמדפיסה את המספרים מ-1 עד 10. בדוגמה זו רואים שימוש ב-BEEP להשמעת צליל קצר, וב-Exit For לביצוע יציאה לא מסודרת לפי רצון המשתמש.

תוכנית B10:

```
Function B10()
Dim Outer As Integer, In1 As Integer
For Outer = 5 To 1 Step -1
   For In1 = 1 To Outer
   Debug.Print In1; ' View Immediate shows results
   Next In1
   Debug.Print    'to go to new line
Next Outer
End Function
```

דוגמה שמציגה For מקונן, כלומר For פנימי בתוך For חיצוני. For חיצוני מגדיר ערך התחלתי 5 למשתנה OUTER, כאשר For פנימי מדפיס את כל המספרים מ-1 עד 5. אחר For חיצוני מחסר 1 (Step -1) ממשתנה OUTER והערך משתנה ל-4. For פנימי מדפיס את כל המספרים מ-1 עד 4. התהליך ממשיך עד שערך Outer = 1 ואז הלולאה הפנימית מדפיסה את הערכים מ-1 עד 1, דהיינו, רק ערך אחד.

לולאה ללא גבול עליון

על אף היעילות והנוחות של השימוש בפקודת For, לא תמיד היא עדיפה על הפקודה Do While. למשל, בתוכנית לחישוב משכורות עובדים, אם לא ידוע מספר העובדים בזמן כתיבת התוכנית, אי אפשר להשתמש בפקודת For. שיטה אחת לטפל במקרה כזה היא להגדיר דגל (Flag) שיקבל על עצמו את הערך 1-. מזינים את שיעור השכר לשעה של כל עובד. כאשר מזינים את הערך 1- עבור השיעור, זה מסמן לתוכנית שהגענו לסוף הנתונים, ואנו רוצים לצאת מלולאת העיבוד.

תוכנית B11:

```
Public Function B11()
' Read input data until the input rate equals the flag
' Requires using the input statement twice, before the loop and in the loop
Const EndOfData = -1   ' Serves as a flag
Dim Rate As Single, Hours As Single, Salary As Single
Rate = InputBox("מה שיעור השכר לשעה?")
Do While Rate <> EndOfData
  Hours = InputBox("כמה שעות עבדת?")
  Salary = Hours * Rate
  MsgBox "המשכורת שלך היא: " & Format(Salary, "$#.00")
  Rate = InputBox("מה שיעור השכר לשעה?")
Loop
End Function
```

במקרה זה קוראים פרטי עובדים עד שמגיעים לעובד האחרון, מזינים בשבילו ערך 1- (EndOfData) עבור השיעור (Rate). ערך זה מוציא מהלולאה, והתוכנית מסתיימת.

דרך אחרת לטפל במקרה שמספר הנתונים לא ידוע מראש, היא להגדיר משתנה מיוחד שישמש כדגל. לאחר כל פעם שמעבדים נתוני עובד, שואלים אם יש כוונה להמשיך. אם התשובה שלילית, מגדירים את ערך המשתנה שמשמש כדגל כ-1- (EndOfData).

תוכנית B12:

```
Public Function B12()
' Read input data until the flag variable indicates end of data
' Requires defining a special flag variable
Const EndOfData = -1   ' Serves as a flag
Dim Rate As Single, Hours As Single, Salary As Single
Dim Flag As Integer
Do While Flag <> EndOfData
  Rate = InputBox("מה שיעור השכר לשעה?")
  Hours = InputBox("כמה שעות עבדת?")
  Salary = Hours * Rate
```

```
  MsgBox "המשכורת שלך היא: " & Format(Salary, "$#.00")
  If (MsgBox("האם סיימת?", vbYesNoCancel) = vbYes) Then
     Flag = EndOfData
 Else
     Flag = 0
End If
Loop
End Function
```

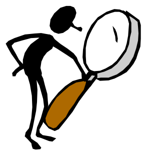
אם המשתמש לוחץ על כן כאשר מופיעה תיבת ההודעה (MsgBox), התוכנית מגדירה את ערך המשתנה Flag ל-1- והלולאה אינה מתבצעת בפעם הבאה.



תרגיל 6:

הדפס רשימה שתכיל את N, N^2,   וסכום כל המספרים השלמים מ-1 עד N עבור 1<=N<=25. הפלט ייראה כך:

N	N^2		1 עד N
1	1	1	1
2	4	1.414	3
.	.	.	.
.	.	.	.



הערה:

כדי להציג את המספרים בעמודות, טוב להשתמש בפקודת TAB(n) שקובעת שהביטוי הבא יופיע בעמודה n. למשל, אם נכתוב:

```
Debug.print tab (10); a; tab(20); b
```

המשתנה a יופיע בעמודה 10 והמשתנה b יופיע בעמודה 20.

תרגיל 7:

כתוב שיגרה שמקבלת כקלט את המספר N ומדפיסה !N (N עצרת). (לדוגמה: !4 שווה 1*2*3*4).

תרגיל 8:

הרחב את השיגרה בתרגיל 7 כך שהיא מקבלת מספר לא קבוע מראש של מספרי N, ומחשבת לכל אחד את !N.

תרגיל 9:

כתוב שיגרה שמוצאת את השלם הגדול ביותר, שהריבוע שלו קטן מ-142,555.



תרגיל 10:

בסופרמרקט מסוים נותנים הנחה 5% על כל קנייה שסכומה מעל לסכום שמוגדר כ"נקודת הנחה". כתוב שיגרה שקוראת רשימת מספרים אשר הראשון בהם מציין את נקודת ההנחה, ושאר המספרים מציינים את מחיר המוצרים שנקנו. ציין בפלט אם לקונה הגיעה הנחה, ומהו הסכום הסופי לתשלום ללא הנחה, ועם הנחה.

תרגיל 11:

חזור על התרגילים 7-10 תוך שימוש בפקודת IF במבנה בסיסי, והנח שסדרת נתוני הקלט אינה קבועה מראש, ולא רק במקרה אחד.

תרגיל 12:

כתוב שיגרה שמקבלת כקלט ספרה A ומספר שלם B. הפלט הוא משתנה בוליאני שערכו TRUE, אם הספרה A היא אחת מהספרות שמרכיבות את המספר השלם B.

תרגיל 13:

הרחב את השיגרה בתרגיל 12 כך שהיא מקבלת מספר לא קבוע מראש של זוגות מספרים A ו-B ומוציאה פלט מתאים לכל זוג.

תרגיל 14:

כתוב תוכנית אשר מקבלת כקלט סדרת זוגות מספרים. המספר הראשון הוא ציון התלמיד בהיסטוריה והשני הוא ציון אותו תלמיד במתמטיקה. הפלט יכלול שני ערכים: מספר התלמידים שקיבלו ציון גבוה יותר בהיסטוריה מאשר במתמטיקה, ומספר התלמידים שקיבלו ציון גבוה יותר במתמטיקה מאשר בהיסטוריה.

תרגיל 15:

כתוב שיגרה שקוראת תמליל באנגלית ומדפיסה אותו כפלט, כך שכל אות מוחלפת על ידי האות שבאה אחריה באלפאבית: האות C מוחלפת באות D וכולי ו-Z מוחלפת על ידי A.

תרגיל 16:

קלט תוכנית שירות לבורסה מכיל 3 נתונים, למשל:

שם מניה	מחיר היום	מחיר אתמול
פיתוח לייזר	139	137
תעשיית מחשבים	2431	2387

כתוב שיגרה שמדפיסה רשימת המניות שעלו או ירדו ביותר מ-10%.



תרגיל 17:

בכתיב אנגלי, הכלל הוא שהאות I קודמת לאות E, חוץ ממקרה שזוג זה בא אחרי האות C, ואז הסדר הוא E לפני I. כתוב שיגרה שקוראת תמליל ובודקת אם כלל זה מתקיים. אם אינו מתקיים, השיגרה מדפיסה את המילה השגויה.

תרגיל 18:

קרא משפט אשר מסתיים בנקודה (אבל ייתכן נקודות אחרות במשפט), ותכנן את הפעולות הבאות:

- מצא את מקום הרווח הראשון, והכנס אותו למשתנה בשם: RevachRishon.
- מצא את האות הראשונה במילה האחרונה, והכנס אותה למשתנה בשם MilaAcharona.
- ספור את המילים במשפט.

תרגיל 19:

כתוב שיגרה שקוראת ומדפיסה תמליל (כמחרוזות, כאשר כל שורה היא מחרוזת), ומחשבת את הערכים הבאים:

- מספר השורות בתמליל
- מספר המשפטים
- מספר המילים

אפשר להניח שכל משפט מסתיים בנקודה ורווח וזהו השימוש היחיד של זוג תווים זה. אפשר גם להניח שמילה מורכבת אך ורק מאותיות.

הסתעפויות חד כיווניות

```
GoTo {label| linenumber}
```

דוגמה:

```
GoTo skipover skipover:
```

הפקודה מעבירה לשורה בעלת התווית.

תוכנית B13: שימוש בפקודה On Error GoTo labelname/linenumber.

```
Public Function B13() 'example of on error goto.
'                 put in bad data and see what happens
 Dim QtySpecial As Integer, Bonus As Currency
 On Error GoTo ErrorMessage
 QtySpecial = InputBox("Enter quantity of special items sold", "Special Sales", "0")
 If QtySpecial Then  'Equivalent to saying QtySpecial <> 0, as in C
     Bonus = 500
  Else
     Bonus = 0
End If
' The message function takes only string or variant
' Format changes its input to variant, Format$ to string
 MsgBox Format$(Bonus, "$#.00"), 65, "Calculated Bonus"
Exit Function
ErrorMessage:
   MsgBox "לא הזנת ערך מספרי" & Chr$(13) & _
   " :מספר הטעות" & Err.Number & Chr$(13) & _
   " :תאור הטעות" & Err.Description
End Function
```

כאשר מתרחשת שגיאה, בעזרת פקודה זו התוכנית מסתעפת לתווית מסוימת (ErrorMessage) בשיגרה. הצגנו הודעה מתאימה וברורה יותר מההודעה המופקת אוטומטית. בנוסף, הצגנו מספר שגיאה (Err.Number) ותיאורה (Err.Description). בסוף גוף השיגרה, הוספנו את הפקודה Exit Function. פקודה זו מוציאה אותנו מהשיגרה לפני הקטע הנקרא ErrorMessage, המיועד לביצוע רק במקרה שגיאה. אפשר לנסות את הפונקציה הזו על ידי הזנת נתונים לא נומריים ל-InputBox.

פקודת Select

פקודה זו היא הרחבה של מבנה IF. IF בסיסי מיועד לשתי תוצאות אפשרויות, כאשר האפשרות הראשונה בולטת יותר. מבנה Select מטפל במספר אפשרויות, כאשר הטיפול בכולן מתבצע במקביל.

המבנה הכללי

```
Select Case <ביטוי לבדיקה>
  [Case <רשימה להשוואה>
  פקודות]
  .....
  [Case Else
  פקודות]
End Select
```

ערך הביטוי לבדיקה חייב להיות ערך שלם, כולל ערך בוליאני ששווה ל-0 (שגוי), -1 (נכון)) או מחרוזת.

רשימה להשוואה היא אחת או יותר מהפריטים הבאים, מופרדים על ידי פסיק:

- ביטוי
- ביטוי TO ביטוי
- ביטוי <אופרטור> IS, כאשר האופרטור הוא אחד מהיחסים:

  {>=|<=|<|>|<>|=}

תוכנית B14:

```
Function B14()
Dim Age As Integer
Age = InputBox("בן כמה אתה?")
Select Case Age
Case 1
  MsgBox"אתה רק יכול לרכב באופניים!"

Case 2, 3, 4
  MsgBox "אתה רק יכול לרכב באופניים!"

Case 5, 6, 7 To 13
  MsgBox "אתה רק יכול לרכב באופניים!"

' Alternative way of showing this
' Case Is < 14
' MsgBox "אתה רק יכול לרכב באופניים!"
Case Is < 16
  MsgBox "אתה רק יכול לרכב באופנוע!"
Case Else
  MsgBox "אתה יכול לנהוג ברכב!"
End Select

End Function
```

בדוגמה זו הצגנו מספר דרכים להצגת תנאי Case. לפי הגיל נקבע אם מותר לרכב באופניים, באופנוע או ברכב. בביצוע, ייבחר רק Case אחד, הראשון שמתקיים.

תוכנית B15:

```
Function B15()
Dim Num1, Num2 As Integer
Dim Op, Ans As String
Num1 = InputBox("Please type a number")
Num2 = InputBox("Please type another number")
Op = InputBox("Do you want +, -, *, or / ?")
Op = UCase$(Op)  'Translates literal operators to capital letters.
Select Case (Op)
Case "+", "ADD", "PLUS"
  Ans = Str$(Num1 + Num2)
  MsgBox Str$(Num1) + " plus" + Str$(Num2) + " is " + Ans
Case "-", "SUBTRACT", "MINUS"
  Ans = Str$(Num1 - Num2)
  MsgBox Str$(Num1) + " minus" + Str$(Num2) + " is " + Ans
Case "*", "MULTIPLY", "TIMES"
  Ans = Str$(Num1 * Num2)
  MsgBox Str$(Num1) + " times" + Str$(Num2) + " is " + Ans
Case "/", "DIVIDE", "INTO"
  Ans = Str$(Num1 / Num2)
  MsgBox Str$(Num1) + " into" + Str$(Num2) + " is " + Ans
Case Else
   Beep
   MsgBox "You didn't answer a correct operator.", 48
End Select
End Function
```

תוכנית זו מחקה מחשב כיס על ידי קליטת שני מספרים, אופרטור אחד וחישוב התוצאה המתאימה.

תוכנית B16:

```
Function B16()
Dim Day As Integer
Dim Error As Boolean
Dim Rate As Single, Hours As Single, Salary As Single
Rate = InputBox("מה שיעור השכר לשעה?")
Hours = InputBox("כמה שעות עבדת?")
Day = InputBox("איזה יום בשבוע (1..7)?")
Select Case Day
Case 1 To 5
  If Hours <= 8 Then
    Salary = Rate * Hours
```

```
  Else
    Salary = Rate * 8 + Rate * 1.5 * (Hours - 8)
  End If
Case 6
  If Hours <= 5 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 5 + Rate * 2 * (Hours - 5)
  End If
Case 7
    Salary = Rate * 3 * Hours
Case Else
  Error = True
End Select
If Error Then

  MsgBox "לא הזנת ערך בין 1 ל-7 עבור היום"
Else
  MsgBox "המשכורת שלך היא: " & Format(Salary, "$#.00")
End If

End Function
```

ביום רגיל השכר הוא 150% לכל שעה נוספת (מעל 8 שעות), ביום שישי השכר הוא 200% לכל שעה נוספת (מעל 5 שעות) ובמוצאי שבת השכר הוא 300% לכל שעת עבודה.

תוכנית B17:

```
Function B17()
Dim Day As String
Dim Error As Boolean
Dim Rate As Single, Hours As Single, Salary As Single
Rate = InputBox("מה שיעור השכר לשעה?")
Hours = InputBox("כמה שעות עבדת?")
Day = InputBox_
("איזה יום בשבוע (ראשון,שני, שלישי, רביעי, חמישי, שישי, או מוצאי שבת)?")
Select Case Day
Case "ראשון", "שני", "שלישי", "רביעי", "חמישי"
If Hours <= 8 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 8 + Rate * 1.5 * (Hours - 8)
  End If
```

```
Case "שישי"
  If Hours <= 5 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 5 + Rate * 2 * (Hours - 5)
  End If
Case "מוצאי שבת"
     Salary = Rate * 3 * Hours
Case Else
     Error = True
End Select
If Error Then
  MsgBox "לא הזנת אחד מן הערכים" & _
  "ראשון, שני, שלישי, רביעי, חמישי, שישי, מוצאי שבת"
Else
  MsgBox "המשכורת שלך היא: " & Format(Salary, "$#.00")
End If

End Function
```

כמו תוכנית B16, רק שהמשתנה עבור פקודת Select הוא מסוג String.

תוכנית B19:

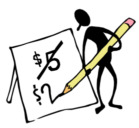
```
Function B19()
Dim Day As Integer
Dim Error As Boolean
Dim Children As Boolean
Dim Rate As Single, Hours As Single, Salary As Single
Rate = InputBox("מה שיעור השכר לשעה?")
Hours = InputBox("כמה שעות עבדת?")
Day = InputBox("איזה יום בשבוע (1..7)?")
Children = InputBox("האם יש לך ילדים למטה מגיל 5 - TRUE או FALSE?")
If Day >= 1 And Day <= 5 And Not Children Then
  If Hours <= 8 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 8 + Rate * 1.5 * (Hours - 8)
  End If
ElseIf Day >= 1 And Day <= 5 And Children Then
  If Hours <= 8 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 8 + Rate * 1.75 * (Hours - 8)
  End If
```

```
ElseIf Day = 6 And Not Children Then
  If Hours <= 5 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 5 + Rate * 2 * (Hours - 5)
  End If
ElseIf Day = 6 And Children Then
  If Hours <= 5 Then
    Salary = Rate * Hours
  Else
    Salary = Rate * 5 + Rate * 2.25 * (Hours - 5)
  End If
ElseIf Day = 7 And Not Children Then
    Salary = Rate * 3 * Hours
ElseIf Day = 7 And Children Then
    Salary = Rate * 3.25 * Hours
Else
  Error = True
End If

If Error Then
  MsgBox "לא הזנת ערך בין 1 ל-7 עבור היום"
Else
  MsgBox " :המשכורת שלך היא" & Format(Salary, "$#.00")
End If

End Function
```

בדוגמה זו השתמשנו בפקודת ElseIf. פקודה זו מאפשרת מספר משתנים לא מוגבל, לעומת פקודת Select, המוגבלת לביטוי אחד בלבד. בדוגמה, אם לעובד יש ילדים, התעריף עבור שעות נוספות גדול ב-25% מהתעריף במקרה שאין ילדים.



תרגיל 20:

כאשר המחשב מוציא רישיון נהיגה, הוא בודק את גיל הנהג. אם הוא בטווח 18-30, מודפס על הרישיון שיש להיזהר מאוד מנסיעה מעל המהירות המותרת. אם הגיל בטווח 31-50, מודפסת הערה שיש לחדש את הרישיון כל 3 שנים. כשהגיל בטווח 51-60, מודפסת הערה שיש לחדש את הרישיון כל שנתיים, ואם הנהג הוא מעל לגיל 60, מודפסת הערה שיש לחדש את הרישיון כל שנה ולעבור בדיקה רפואית כל שנתיים. הכן שיגרה שתקלוט תאריך לידה של הנהג, ותדפיס את ההודעה המתאימה.



תרגיל 21:

העלות למשלוח חבילות דואר בארץ מבוססת על 3 קריטריונים:

1. מקום היעד:
   - M - מקומי
   - E - עיר גדולה (ירושלים, ת"א, חיפה, באר שבע)
   - K - עיר קטנה, מושב, קיבוץ, וכולי
2. משקל:
   - K - קל (פחות מקילו)
   - M - ממוצע (בין קילו ל-10 קילו)
   - C - כבד (למעלה מ-10 קילו)
3. שירות:
   - R - רגיל
   - E - אקספרס

התעריף נקבע לפי נקודות, שנקבע להן ערך שקלי. הטבלה הבאה מאפשרת לחשב את המחיר למשלוח. בחלק העליון של הטבלה מופיעות כל התמורות השונות של יעד, משקל וסוג שירות. בחלק התחתון נרשם התעריף בנקודות. כתוב שיגרה שמקבלת כקלט את מקום היעד (M, E, K), משקל בקילו, סוג השירות (E, R) והערך השקלי של "נקודה". השיגרה תפיק כפלט את המחיר למשלוח.

יעד	M	M	M	M	M	M	E	E	E	E	E	E	K	K	K	K	K	K
משקל	K	K	M	M	C	C	K	K	M	M	C	C	K	K	M	M	C	C
שירות	E	R	E	R	E	R	E	R	E	R	E	R	E	R	E	R	E	R
6 נקודות		X						X										
12 נקודות	X						X						X	X				
50 נק' + 2 לקילו מעל ל-20						X												X
100 נק' + 2 לקילו מעל ל-20					X												X	
2 נק' לקילו				X														
3 נק' לקילו			X							X		X						
6 נק' לקילו									X		X					X		
8 נק' לקילו															X			



תרגיל 22:

במשחק מזל, אדם מהמר X שקלים ובוחר פתק. אם הפתק ירוק הוא מפסיד את כספו. אם הפתק חום הוא זוכה ב-150%, ואם הפתק ירוק הוא מקבל 300%. כתוב שיגרה שתקלוט את סכום ההימור וצבע הפתק שנבחר, ותוציא כפלט את סכום התמורה שהאדם זכאי לה.

תרגיל 23:

קיימים נתונים לגבי צבע השיער (שחור, חום, ג'ינג'י, בלונדיני) וצבע העיניים (כחול, חום, ירוק) של אוכלוסיה כלשהי, שגודלה לא ידוע מראש. כתוב שיגרה שתוציא כפלט את גודל האוכלוסיה, ואת אחוז האנשים שיש להם כל צירוף של שיער ועיניים, למשל אחוז האנשים בעלי שיער חום ועיניים ירוקות.

3 פרק

מערכים

מערכים במימד אחד: וקטור

מנהל עבודה מודאג שהפועלים מבלים בסוף השבוע, ומאחרים לעבודה ביום ראשון. כדי לתעד את טענתו, הוא רוצה לראות אם סכום שעות העבודה הטיפוסי של שעות העבודה של כל הפועלים יחד בימים ב׳ עד ה׳ גבוה יותר מסכום שעות העבודה ביום א׳.

תוכנית C01:

```
Public Function C01()
Dim Hours1 As Single, Hours2 As Single, Hours3 As Single, Hours4 As Single, Hours5  _
        As Single
Dim Sun As Single, Mon As Single, Tues As Single, Wed As Single, Thurs As Single
Dim Counter As Integer
For Counter = 1 To 2
1:   Hours1 = InputBox("כמה שעות עבדת ביום ראשון?")
2:   Hours2 = InputBox("כמה שעות עבדת ביום שני?")
3:   Hours3 = InputBox("כמה שעות עבדת ביום שלישי?")
4:   Hours4 = InputBox("כמה שעות עבדת ביום רביעי?")
5:   Hours5 = InputBox("כמה שעות עבדת ביום חמישי?")
6:   Sun = Sun + Hours1
7:   Mon = Mon + Hours2
8:   Tues = Tues + Hours3
9:   Wed = Wed + Hours4
10:  Thurs = Thurs + Hours5
Next
MsgBox "יום ראשון סכום השעות: " & Sun & Chr(13) & _
       "יום שני סכום השעות: " & Mon & Chr(13) & _
       "יום שלישי סכום השעות: " & Tues & Chr(13) & _
       "יום רביעי סכום השעות: " & Wed & Chr(13) & _
       "יום חמישי סכום השעות: " & Thurs & Chr(13)
End Function
```

בתוכנית זו יש ריבוי משתנים וריבוי פקודות. יש הקבלה בין המשתנים Hours1 עד Hours5 המתבטאת בפקודות 1-5. גם המשתנים Sun עד Thurs מקבילים, כפי שנראה בפקודות 6-10. פקודות אלו זהות כמעט, כאשר ההבדלים היחידים ביניהן הם השינויים שחייבים לבצע כדי שאותה פקודה תטפל ביום אחר בשבוע.

ניתן לפתור את בעיית ריבוי המשתנים אם ניתן שם אחד לכל קבוצת משתנים. למשל, לקבוצה Hours1 עד Hours5 נוכל לקרוא Hours ולהבדיל בין משתנה למשתנה בתוך הקבוצה על ידי הוספת מציין (Subscript) בסוגריים. בהתאם לכך, המשתנה Hours הוא משתנה רב-ערכי או **מערך** (Array), כי הוא מורכב מ: (1) Hours, Hours(2), Hours(3), Hours(4), ו-Hours(5).

עתה, נוכל לכתוב Hours(I) כדי לתאר איבר כלשהו של המערך Hours. נתייחס למכלול הערכים במערך זה אם ניתן ל-I, בזה אחר זה, את תחום הערכים מ-1 עד 5. נוכל לתאר את המערך Hours גם כגוש כתובות (תאים) רצופות בזיכרון המחשב, כפי שמוצג בתרשים הבא.

Hours(1)	Hours(2)	Hours(3)	Hours(4)	Hours(5)
17	13	3	34	1

גם את המשתנים Sun עד Thurs נוכל לקבץ כקבוצה (מערך) בשם Days. אם נשתמש במציין J כדי לתאר איבר כלשהו במערך, נוכל לכתוב Days(J).

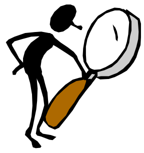
לאחר ההגדרות שכתבנו, כל אחד מהשמות Hours ו-Days מייצג קבוצת ערכים הסדורים בזה אחר זה במערך חד-מימדי, אשר נכנה אותו **רשימה** (List) או **וקטור** (Vector).

תוכנית C02: שיבוץ המערכים Hours **ו-** Days בתוכנית הקודמת כדי לקבל תוכנית משופרת.

```
Public Function C02()
Dim Hours(1 To 5) As Single
Dim Days(1 To 5) As Single
Dim Counter1 As Integer, Counter2 As Integer

For Counter1 = 1 To 2
  For Counter2 = 1 To 5
     Hours(Counter2) = InputBox("כמה שעות ביום עבדת " & Counter2)
     Days(Counter2) = Days(Counter2) + Hours(Counter2)
  Next
Next
For Counter1 = 1 To 5
   MsgBox  "סכום השעות ביום: " & Counter1 & "שווה ל:" & Days(Counter1)
 Next
End Function
```

כשם שמצהירים על משתנים, כך יש להצהיר על מערכים. בהצהרת המשתנים מציינים שכל אחד מהשמות Hours ו-Days הוא מערך בן 5 איברים ממשיים מסוג Single. טווח המציינים של כל אחד מהמערכים הוא מ-1 עד 5, כפי שמופיע בסוגריים. כל איברי מערך אחד חייבים להיות מאותו סוג, שהרי יש רק מילת תיאור סוג אחת (Single במקרה זה) עבור כולם. שים לב ששתי פקודות For מחליפות את הפקודות בשורות 1-10 בגירסה הקודמת של התוכנית. אם הרווח אינו נראה גדול, תאר את המצב לו רצינו לחבר נתונים עבור 30 יום ולא 5 ימים. מספר השורות בתוכנית C01 היה גדל ל-60, כאשר מספר השורות בתוכנית C02 היה נשאר כמות שהוא.



הערה:

אם לא נציין בפקודת Dim את טווח המציינים (למשל (Hours (1 to 5), אלא נציין רק (5) Hours, המהדר יבין שאנו רוצים להגדיר 6 משתנים בווקטור, ממוספרים מ-0 עד 5. אם כן נציין טווח, נוכל לבחור כל ערך שלם, כולל מספרים שליליים, למשל: 6- To 13- עבור קצוות הטווח.



תרגיל 1:

כתוב שיגרה שמקבלת 5 איברים של וקטור כקלט, ומחליפה כל איבר בסכום כל האיברים הקודמים לו הכולל גם אותו איבר עצמו.

תרגיל 2:

כתוב שיגרה שקוראת וקטור לא מסודר של 10 ערכים שלמים חיוביים, כאשר חלק מהערכים חוזרים על עצמם. הפלט יהיה אותו וקטור באותו סדר, כאשר כל ערך מופיע בדיוק פעם אחת. כל ערך החוזר על עצמו יוחלף ב-0. למשל, אם הווקטור המקורי הוא (מימין לשמאל):

15 3 14 9 15 8 17 8 15 7

וקטור הפלט יהיה:

0 3 14 9 0 0 17 8 15 7

תרגיל 3:

בתרגיל הקודם, קבץ את כל הערכים הייחודיים בהתחלת הווקטור, והצב 0 במקומות האחרונים של הווקטור, מבלי לשנות את סדר הערכים. בנוסף, הוצא כפלט את משתנה NumUnique שערכו יהיה שווה למספר האיברים בווקטור המקורי שאינם חוזרים על עצמם. למשל, אם הווקטור המקורי הוא (מימין לשמאל):

15 3 14 9 15 8 17 8 15 7

וקטור הפלט יהיה:

3 14 9 17 8 15 7

כאשר ערך NumUnique יהיה 7.

טעינת מערכים

לפעמים הנתונים שרוצים לטעון למערך ידועים מראש, כמו נתוני טבלה. למשל, במקרה של שכר עובדים, כדי לקבל את השכר עבור שעות נוספות ביום מסוים, משתמשים בנוסחה הבאה: ShiurShaotNosafot * Sachar * MisparShaot. השכר (שיעור) לשעות נוספות שונה מיום ליום. כמובן, יכולנו להגדיר מערך ולטעון את הערכים אחד אחד, כדלהלן:

```
Dim Shiur (1 to 7)
Shiur (1) = 1.5
Shiur (2) = 1.5
Shiur (3) = 1.5
Shiur (4) = 1.5
Shiur (5) = 1.5
Shiur (6) = 2
Shiur (7) = 3
```

דרך יותר קצרה היא להשתמש במשתנה מסוג Variant, יחד עם הפונקציה החדשה Array, שטוענת את כל המערך לתוך משתנה זה, ומאפשרת קריאת הערכים במערך בצורה המקובלת. תוכנית C021 מדגימה שיטה זו.

תוכנית C021:

```
Dim Shiur ` Default is Variant
Shiur = Array (1.5, 1.5, 1.5, 1.5, 1.5, 2, 3)
For I = 1 to 7
   Debug.Print Shiur (I)
Next
```

מערכים דינמיים

מערך דינמי הוא מערך שמספר איבריו יכול להשתנות, משום שאיננו יודעים מראש לכמה איברים נזדקק בהמשך התוכנית. היתרון העיקרי במערך דינמי הוא שאין צורך להגדיר מערכים גדולים כאשר בסופו של דבר איננו משתמשים ברוב האיברים, ולכן המקום שהוקצה להם הולך לאיבוד. השלבים בהגדרת מערך דינמי הם כדלהלן:

1. הגדר מערך ללא מימדים, למשל:

```
Dim TestArray () as Integer
```

2. בשלב שתוכל לקבוע את גודל המערך, השתמש בהצהרה החדשה: ReDim כדלהלן:

```
ReDim TestArray (4 To 16)
```

שים לב שבשלב זה אסור להגדיר את סוג הנתונים, אותו חייבים להגדיר בפקודת Dim המקורית. הגבולות התחתונים והעליונים אינם חייבים להיות ערכים קבועים. הם יכולים להיות משתנים שנקראו בתוכנית.

3. לאחר שהגבולות הוגדרו פעם אחת, ניתן להגדירם שוב בהמשך בעזרת פקודת ReDim. הבעיה העיקרית היא שהנתונים הקיימים במערך יאבדו כשנגדיר את המערך בפעם השנייה, אלא אם מוסיפים את המילה Preserve, כדלהלן:

```
ReDim Preserve TestArray (1 To Up)
```

שימוש במערך חד-מימדי

נניח שמורה בודק מבחן ורוצה לתת לסטודנטים ציונים יחסיים. לציון הנמוך ביותר הוא קובע ערך 0 ולציון הגבוה ביותר הוא קובע ערך 100. כל שאר הציונים יחושבו באופן יחסי בתוך תחום זה. למשל, אם הציון הגבוה ביותר הוא 72, ייקבע לסטודנט זה ציון 100. לסטודנט שקיבל את הציון הנמוך ביותר - 63, יינתן ציון 0. סטודנטים שציוניהם בתוך התחום 63 עד 72 יקבלו ציונים בין 63 ו-72, מפוזרים באופן יחסי למרחק שלהם מ-63. הנוסחה המתאימה לחישוב ציון יחסי היא:

ציון יחסי = 100 * **(ציון – ציון נמוך)\(ציון גבוה – ציון נמוך)**

תוכנית C03:

```
Public Function C03()
Dim Grades() As Integer
Dim EndOfData As Boolean
Dim High As Integer, Low As Integer
Dim Index As Integer
Dim Counter As Integer
Dim sngFinalGrade As Single
High = 0: Low = 100
Do Until EndOfData
   Index = Index + 1
   ReDim Preserve Grades(1 To Index)
   Grades(Index) = InputBox("הזן ציון של תלמיד")
   If Grades(Index) > High Then High = Grades(Index)
   If Grades(Index) < Low Then Low = Grades(Index)
   EndOfData = InputBox("האם הגענו לסוף הנתונים (True או False) ")
Loop
For Counter = 1 To Index
    sngFinalGrade = 100 * (Grades(Counter) - Low) / (High - Low)
    MsgBox "הציון היחסי של תלמיד מספר " & Counter  & " הוא: " & _
        Format(sngFinalGrade, "#0.00")
Next
End Function
```

בהתחלה, הציון הגבוה רשום כ-0 והנמוך כ-100. בכל קריאת נתונים, מעדכנים את האינדקס העליון של המערך, שהרי הוא דינמי. לאחר קריאת הנתון, מעדכנים את הציון הגבוה והנמוך ביותר, ומכניסים את המספר החדש למערך.



תרגיל 4:

חזור על תרגילים 1-3, כאשר גודל הווקטור אינו ידוע מראש, כלומר השיגרה תעבוד נכון עבור וקטור בגודל כלשהו.

תרגיל 5:

כתוב שיגרה שמקבלת כקלט נתונים על כמות הגשמים הממוצעת בכל חודש, ומפיקה כפלט:

- כמות גשמים ממוצעת בשנה.
- החודש הגשום ביותר.
- החודש השני הגשום ביותר.
- החודש שבו ירדה כמות הגשמים הקטנה ביותר.
- ההפרשים בין כמויות הגשמים שירדו בכל חודש לבין כמות הגשמים הממוצעת בכל החודשים.

בעיית הפלינדרום

פלינדרום (Palindrome) הוא כינוי למשפט, אשר אפשר לקרוא אותו משני כיוונים ולקבל אותה תוצאה, בתנאי שמתייחסים רק לאותיות ומתעלמים מרווחים ומתווי פיסוק. למשל המשפט : Madam, I'm Adam הוא פלינדרום.

תוכנית C04:

```
Public Function C04()

Dim Str As String
Dim Substring As String
Dim start As Integer, End1 As Integer

Dim KeyWord As String
Dim Length As Integer
Dim Location As Integer, Loc As Integer

Dim Letters() As String
Location = 1
Str = InputBox("בבקשה להזין את המחרוזת לבדיקה"," Palindromes", "")
Length = Len(Str)
Substring = Mid$(Str, Location, 1)
Do While Location <= Length
   If Substring >= "a" And Substring <= "z" Then ' accepts small and big letters
        Loc = Loc + 1
```

```
        ReDim Preserve Letters(1 To Loc)
        Letters(Loc) = Substring
    End If
        Location = Location + 1
        Substring = Mid$(Str, Location, 1)
Loop
start = 1
End1 = Loc
Do While start < End1 And Letters(start) = Letters(End1)
  start = start + 1
  End1 = End1 - 1
Loop
If start >= End1 Then
  KeyWord = " is"
Else
  KeyWord = " isn't"
End If
MsgBox Str & KeyWord & " a Palindrome"

End Function
```

בחלק הראשון של התוכנית, קוראים את המחרוזת לתוך מערך דינמי שנקרא Letters. מעבירים למערך את התווים החוקיים ללא פיסוק. בחלק השני, משווים את התווים שהם במקום סימטרי בהתחלה ובסוף המערך כדי לראות אם הם זהים. אם התשובה חיובית, המשפט מוצהר כפלינדרום, אחרת, מודיעים שהוא אינו פלינדרום.



תרגיל 6:

כתוב שיגרה שקוראת רשימת שמות ומדפיסה אותם בסדר הפוך.

תרגיל 7:

כתוב שיגרה שמקבלת כקלט שורות תמליל באנגלית, מכניסה אותן למערך ומדפיסה. אם צירוף התווים "^N#" מופיע בתחילת השורה, היא תודפס בכניסה של N מקומות מימין, כאשר N הוא מספר בין 1 ל-120.

תרגיל 8:

השיגרה הבאה מטפלת באיתור פושעים לפי שתי קבוצות נתונים:

- כרטסת 5 הפושעים הידועים למשטרה, שכוללת את הנתונים: שם, גובה, ומשקל. אפשר לקרוא נתונים אלה למחשב מכל מקור חיצוני אחר, או אפשר לקבוע את ערכם כחלק מהשיגרה.
- תיאור הפשע, הגובה והמשקל המשוערים של הפושע.

כתוב שיגרה שתדפיס את תיאור הפשע ואת שמות החשודים בביצועו. החשודים הם אנשים שגובהם חורג בפחות מ-5 ס"מ מהגובה המשוער ומשקלם חורג בפחות מ-5 קילו מהמשקל המשוער של מבצע הפשע.

משתנים רב-ערכיים בשני מימדים

המערכים **Hours** ו-**Days** מייצגים ערכים סדורים בזה אחר זה במערך חד-מימדי, במבנה **רשימה** או **וקטור**. קיימות קבוצות ערכים המסודרים במבנה **מערך דו-מימדי** (Two-dimensional Array). מערך זה נקרא **טבלה** (Table) או **מטריצה** (Matrix).

נדגים באמצעות תוכנית את השימוש במערך דו-מימדי. במפעל מסוים קובעים את משכורות עובדי המנהלה כפונקציה של שנות הוותק בתפקיד, ושל רמת ההשכלה. הנתונים ערוכים בטבלה שבהמשך ומייצגים את האחוז שבו יש לכפול את השכר הבסיסי כדי לקבל את השכר לכל עובד.

השכלה	פחות מ-5 שנות ותק	ותק בין 5 ל-10	ותק בין 10 ל-15	יותר מ-15 שנות ותק
בלי בגרות	100	125	150	175
בגרות	125	150	175	200
תואר 1 או יותר	150	175	200	225

תוכנית C05:

```
Public Function C05()
Const BasicSalary = 1000
Dim Salary(1 To 3, 1 To 4) As Integer
Dim EndOfData As Boolean
Dim I As Integer, J As Integer, WorkerNumber As Integer, _
   Education As Integer, Seniority As Integer
Dim FinalSalary As Single
On Error GoTo 9
MsgBox"הכנס את הטבלה לחישוב משכורות"

For Education = 1 To 3
   For Seniority = 1 To 4 'Input along rows
Cont9:        Salary(Education, Seniority) = InputBox("הזן את הערך המתאים לשורה: " &
Education & " ולטור: " & Seniority)
   Next
Next

On Error GoTo 99
Do Until EndOfData
  WorkerNumber = InputBox("הזן את מספר העובד")
Cont99:
  Education = InputBox("הזן קוד עבור השכלה - בין 1 ל-3")
  Seniority = InputBox("הזן קוד עבור ותק - בין 1 ל- 4")
```

```
  FinalSalary = (Salary(Education, Seniority) / 100) * BasicSalary
  MsgBox "השכר הסופי עבור פועל מספר " & WorkerNumber & " הינו: " & FinalSalary
  On Error GoTo 999
Cont999:
  EndOfData = InputBox("האם הגענו לסוף הנתונים (True או False)")
Loop
Exit Function
99: MsgBox "הזנת אינדקס מחוץ לטווח. נסה שוב"
Resume Cont99
9: MsgBox "הזנת ערך לא נומרי. נסה שוב"
Resume Cont9
999: MsgBox "חייבים לכתוב TRUE או FALSE"
Resume Cont999
End Function
```

הצהרת משתנים עבור מטריצה, דומה להצהרת משתנים עבור וקטור. בהצהרת המשתנים דרושים שני זוגות מספרים, כדי להגדיר את תחום הערכים של שני המציינים. הזוג הראשון מתייחס לתחום השינוי של אינדקס השורות בטבלה, והזוג השני מתייחס לתחום השינוי של אינדקס העמודות, כאשר הם מופרדים על ידי פסיק.

שורה Cont9: מציינים איבר בשורה I ועמודה J על ידי Salary (I,J), בדוגמה זו Salary(Education,Seniority). המשתנה Education בלולאה החיצונית מקבל ערכים מ-1 עד 3. כאשר Education = 1, המשתנה Seniority שבלולאה הפנימית משתנה מ-1 ל-4. כלומר, ארבעת המספרים הנקראים תחילה הם: Salary(1,1), Salary(1,2), Salary(1,3), ו-Salary(1,4), הכתובים בשורה הראשונה של הטבלה. אחר כך ערך Education שבלולאה החיצונית עולה ל-2, ושוב, ערך Seniority עולה מ-1 ל-4 בלולאה הפנימית. לבסוף, ערך Education בלולאה החיצונית יהיה 3, ו-Seniority ישתנה מ-1 ל-4 בלולאה הפנימית במהלך קריאת השורה השלישית.

הפקודה On Error מופיעה 3 פעמים, כל פעם לפני שגיאה מסוג אחר. פעם אחת היא מפנה אותנו לשגרת שגיאה שמבקשת שהמשתמש יזין ערך מספרי, פעם לשגרת שגיאה שמבקשת שהמשתמש יזין ערך בטווח מסוים, ופעם לשגרת שגיאה שמבקשת שהמשתמש יזין ערך בוליאני. בכל מקרה, הפקודה Resume Cont9 (למשל) מחזירה למקום השגיאה הספציפית ומאפשרת עוד הזדמנות להזין את הנתונים בצורה נכונה.

תוכנית C06:

```
Public Function C06()
Dim Rows As Integer, Columns As Integer
Dim I As Integer, J As Integer
Dim RowSum() As Integer
Dim ColSum() As Integer
Dim Matrix() As Integer
1: Rows = InputBox("כמה שורות תהיינה במטריצה")
```

```
2: Columns = InputBox("כמה עמודות תהיינה במטריצה")
ReDim Preserve Matrix(1 To Rows, 1 To Columns)
ReDim Preserve RowSum(1 To Rows)
ReDim Preserve ColSum(1 To Columns)
3: For I = 1 To Rows
      For J = 1 To Columns  'Input along rows
        Matrix(I, J) = InputBox("הזן את הערך המתאים לשורה: " & I & " ולטור:  " & J)
      Next
  Next
4: For I = 1 To Rows
      For J = 1 To Columns
          RowSum(I) = RowSum(I) + Matrix(I, J)
          ColSum(J) = ColSum(J) + Matrix(I, J)
      Next
  Next
5: For I = 1 To Rows
      For J = 1 To Columns
         Debug.Print Matrix(I, J);
      Next
      Debug.Print "  ";
      Debug.Print RowSum(I)
  Next
  Debug.Print
6: For J = 1 To Columns
      Debug.Print ColSum(J);
  Next
End Function
```

בתוכנית זו אנו מוצאים את סכום השורות והעמודות במטריצה. התוכנית משתמשת במערכים דינמיים כדי לקבוע את גודל המטריצה, שלא נקבע בזמן כתיבת התוכנית.

שורות 1-2 : קליטת מספר השורות והעמודות במטריצה.

שורה 3 : קליטת תוכן המטריצה לפי שורות.

שורה 4 : בניית סכום השורות והעמודות.

שורה 5 : הדפסת השורות וסכום כל שורה בסוף השורה.

שורה 6 : הדפסת השורה התחתונה המכילה את סכום העמודות.



תרגיל 9:

נתונה מטריצה במבנה **ריבוע קסם** (Magic Square), שבו סכום המספרים שווה בכל שורה, בכל עמודה ובכל אלכסון, למשל:

6	1	8
7	5	3
2	9	4

כתוב שיגרה שמקבלת כקלט את מימדי הריבוע ואת ערכיו, ובודקת אם הוא ריבוע קסם.

תרגיל 10:

ריבוע לטיני הוא מערך של N*N שמכיל מספרים, כך שמספר כלשהו אינו מופיע יותר מפעם אחת בשורה או בעמודה. הריבוע הבא עומד בדרישות אלו:

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

כתוב שיגרה שבודקת אם מערך בגודל N*N הוא ריבוע לטיני.

תרגיל 11:

האחד בינואר 1998 חל ביום חמישי. כתוב שיגרה שמדפיסה את לוח השנה 1998 + N (עבור כל N). פלט לחודש מסוים צריך להיראות כך:

Janurary 1998

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

תרגיל 12:

כתוב שיגרה שמקבלת כקלט מערך דו-מימדי Tziyun (I,K) אשר מכיל את הציון (מ-0 עד 10) של סטודנט I בשאלה K. קלוט גם את הווקטור Mishkal שמחזיק את שקלול שאלה K בחישוב הציון הסופי. כתוב שיגרה שתוציא עבור סטודנט I את הציון המשוקלל שלו.



תרגיל 13:

מוסד לגמילות חסדים מנסה למכן את חלוקת הכספים למשפחות. כתוב שיגרה שתקבע סכום שמקבלת משפחה על פי הטבלה הבאה.

נתוני הקלט הם:

- טבלה לקביעת השכר
- רמת ההכנסה של המשפחה
- מספר הילדים במשפחה

ילדים / הכנסה	1	2	3	4	5	6 ומעלה
עד 10,000	3,000	7,000	12,000	16,000	21,000	25,000 ועוד 4,000 /ילד נוסף
10,000-19,999	2,000	5,000	7,5000	9,000	11,000	13,000 ועוד 2,000 /ילד נוסף
20,000-29,999	1,000	2,800	4,800	6,500	8,000	9,000 ועוד 1,300 /ילד נוסף
30,000-39,999	1,000	1,800	3,000	4,000	5,200	9,000 ועוד 1,300 /ילד נוסף
40,000-49,999	800	1,500	2,500	3,200	3,800	4,500 ועוד 600 /ילד נוסף
50,000-59,999	500	800	1,500	2,100	2,600	3,400 ועוד 400 /ילד נוסף
מעל 60,000	0	0	0	0	0	0

חיפוש במערכים

חיפוש - סריקת כל האיברים במערך כדי למצוא את האיבר, או האיברים, שמקיימים תנאי מסוים, או שווים לערך מסוים. יש שתי שיטות נפוצות לחיפוש איברים במערך. השיטה הפשוטה ביותר היא **חיפוש ליניארי** (Linear Search), החיפוש היעיל ביותר (במיוחד למערכים גדולים) נקרא **חיפוש בינארי** (Binary Search).

לא נחזיק מערכים גדולים ב-Access. אם קיימים נתונים רבים, נחזיק אותם כטבלה (עיין בפרק 5). קיימת פקודת חיפוש על טבלאות, ולכן אין צורך לתכנת את החיפוש. נשתמש במערכים לכמויות קטנות של נתונים, ונוכל להסתפק בחיפוש ליניארי.

חיפוש ליניארי מתחיל באיבר הראשון במערך ומתקדם לסוף, עד איתור האיבר הרצוי.

תוכנית C07:

```
Public Function C07()
Dim PhoneNumbers()
Dim EndOfData As Boolean
Dim ShemChaver
Dim Index As Integer, NumNames As Integer
Dim nimza As Boolean
' Input data in telephone list
Do Until EndOfData
  Index = Index + 1
 '  Note Preserve only functions on the last dimension
  ReDim Preserve PhoneNumbers(1 To 2, 1 To Index)
  PhoneNumbers(1, Index) = InputBox("הזן שם החבר")
  PhoneNumbers(2, Index) = InputBox("הזן מספר טלפון של החבר")
  If (MsgBox("האם סיימת?", vbQuestion + vbYesNoCancel) = vbYes) Then
    EndOfData = True
  Else
    EndOfData = False
  End If
Loop
EndOfData = False
NumNames = Index
' Search for names in the telephone list
Do Until EndOfData
  nimza = False
  ShemChaver = InputBox("הזן שם החבר שמחפשים")
  Index = 1
  Do While (Index <= NumNames And Not nimza)
    If ShemChaver = PhoneNumbers(1, Index) Then
      nimza = True
    Else
      Index = Index + 1
    End If
  Loop

  If nimza Then
   MsgBox " מספר הטלפון עבור" & ShemChaver & "  " & PhoneNumbers(2, Index)
  Else
   MsgBox " מספר הטלפון עבור" & ShemChaver &" אינו ברשימה "
  End If
  If (MsgBox("האם סיימת?", vbQuestion + vbYesNoCancel) = vbYes) Then
    EndOfData = True
  Else
    EndOfData = False
  End If
Loop

End Function
```

בקטע הראשון, קוראים נתונים לתוך רשימת שמות חברים ומספרי הטלפון שלהם, המוחזקים במערך. בעזרת הביטוי Preserve מותר לשנות את ערך המימד האחרון בלבד, ולכן סידרנו שהמימד המשתנה יהיה דווקא המימד השני. בקטע השני, משתמשים בחיפוש ליניארי כדי למצוא שמות חברים מסוימים, ולהוציא כפלט את מספר הטלפון של כל אחד.



תרגיל 14:

בארגון מסוים המשכורת המירבית נקבעת לפי שנות ותק, כפי שמוצג בטבלה. אם שנות הוותק הן מעל 25, או פחות מ-3, אין חובה לנהוג לפי כללים אלה.

שנות ותק	משכורת מירבית
3-5	6,475
6-10	10,578
11-13	16,820
14-20	18,100
21-25	19,500

כתוב שיגרה שמקבלת כקלט שמות עובדים ושנות הוותק שלהם. הפלט יהיה דוח המכיל את שמות העובדים והמשכורת המירבית שלהם על פי שנות הוותק. בסיום יש להדפיס את מספר העובדים בכל קבוצת ותק.

תרגיל 15:

לתוכנית המטפלת בהשבת אבדות, יש שני סוגי קלט:

- תיאור המציאה, שם המוצא, ומספר הטלפון שלו.
- שם המאבד ותיאור האבדה.

אם קיימת התאמה בין תיאור האבדה לבין תיאור המציאה, התוכנית מדפיסה שורה עם הפרטים הבאים:

תיאור האבדה, שם המאבד, שם המוצא ומספר הטלפון שלו.

כתוב שיגרה שקולטת קלט מהסוג הראשון לתוך מערך, ומחפשת את האבדה המתאימה לכל שורת קלט מהסוג השני.

מיון במערכים

מיון - פעולה של סידור איברים לפי תכונה מוגדרת, כמו למשל לפי הערך שלהם. אפשר למיין בשני סדרי מיון (Collating Sequences):

- **סדר עולה** (Ascending Order) - לאחר מיון, האיבר בעל הערך הקטן ביותר יהיה הראשון ברשימה, והאיבר בעל הערך הגדול ביותר יהיה האחרון ברשימה.
- **סדר יורד** (Descending Order) - לאחר מיון, האיבר בעל הערך הגדול ביותר יהיה הראשון ברשימה, והאיבר בעל הערך הקטן ביותר יהיה האחרון ברשימה.

יש מספר שיטות מיון. כפי שהזכרנו, לא נחזיק מערכים גדולים ב-Access, אם יש נתונים רבים נחזיק אותם כטבלה. קיימת פקודת מיון עבור טבלאות, ואין צורך לתכנת את המיון. נשתמש במערכים עבור כמויות קטנות של נתונים, ונוכל להסתפק בשיטה שנקראת מיון על ידי **בחירה ליניארית** (Sorting By Linear Selection).

במיון על ידי בחירה ליניארית, אנו מניחים שקיימים במערך מספר ערכים (למשל, NumNames) שאינם מסודרים. בכדי למיינם, עלינו לבצע את הצעדים הבאים:

1. נציב 1 במשתנה Katan, המיועד להצביע על הערך הקטן ביותר במערך. נשווה את: PhoneNumbers (1,Katan) עם: PhoneNumbers (1,2) ונקבל אחת משתי התוצאות האפשריות הבאות:
   - PhoneNumbers (1,Katan) =< PhoneNumbers (1,2). במקרה כזה, Katan יישאר 1.
   - PhoneNumbers (1,Katan) > PhoneNumbers (1,2). במקרה כזה, Katan יקבל את הערך 2.
2. נשווה את: PhoneNumbers (1,Katan) עם: PhoneNumbers (1,3). אם PhoneNumbers (1,Katan) > PhoneNumbers (1,3), אז Katan יהפוך להיות 3, אחרת הוא יישאר ללא שינוי.
3. נמשיך כך עבור: PhoneNumbers (1,4), PhoneNumbers (1,5), עד שנשווה את: PhoneNumbers (1,NumNames) עם: PhoneNumbers (1,Katan). לאחר כל השוואה, האינדקס Katan יצביע על האיבר שערכו קטן יותר. לאחר ההשוואה האחרונה, האינדקס יצביע על האיבר הקטן ביותר, והמשתנה PhoneNumbers (1,Katan) יכיל את הערך הקטן ביותר מבין כל NumNames הערכים שבמערך.
4. עתה יש להחליף את PhoneNumbers (1,1) ב- PhoneNumbers (1,Katan). כתוצאה מההחלפה, האיבר בעל הערך הקטן מכל האיברים נמצא **במקום הראשון** במערך. כך מסתיים המעבר הראשון (First Pass).
5. המעבר השני דומה לראשון, אלא שמציבים 2 במשתנה Katan, עורכים את החיפוש מ: PhoneNumbers (1,3) עד: PhoneNumbers (1,NumNames), כדי למצוא את האיבר הקטן ביותר מבין 1 - NumNames איברים שאחרי האיבר הראשון. כזכור, האיבר הראשון נבחר במעבר הראשון כבעל הערך הקטן ביותר.

האיבר הקטן ביותר שנמצא עכשיו יוצב לפי ערכו במקום השני במערך, ויהיה השני בסדר המיון.

6. במעבר השלישי, מציבים 3 במשתנה Katan, וממשיכים כמו קודם. במעבר האחרון, Katan יקבל ערך התחלתי של NumNames - 1. נשווה את הערך שנמצא באותו מקום, שהוא השני מהאחרון, לערך שנמצא במקום האחרון, וכאשר נסדר את שני הערכים האחרונים, המערך יהיה מסודר במלואו.

תוכנית C08: מיון על ידי בחירה ליניארית.

```
Public Function C08()
Dim PhoneNumbers()
Dim EndOfData As Boolean
Dim Temp1, Temp2
Dim Index As Integer, Index2 As Integer, NumNames As Integer
Dim Katan As Integer
' Input data in telephone list
Do Until EndOfData
  Index = Index + 1
 '  Note Preserve only functions on the last dimension
  ReDim Preserve PhoneNumbers(1 To 2, 1 To Index)
  PhoneNumbers(1, Index) = InputBox("הזן שם החבר")
  PhoneNumbers(2, Index) = InputBox("הזן מספר טלפון של החבר")
  If (MsgBox("האם סיימת?", vbQuestion + vbYesNoCancel) = vbYes) Then
    EndOfData = True
  Else
    EndOfData = False
  End If
Loop

NumNames = Index

' Print out in original order
Debug.Print
Debug.Print "Original Order"
For Index = 1 To NumNames
  Debug.Print PhoneNumbers(1, Index), PhoneNumbers(2, Index)
Next

' Sort names in the telephone list
For Index = 1 To NumNames - 1
  Katan = Index
  For Index2 = Index + 1 To NumNames
     If PhoneNumbers(1, Index2) < PhoneNumbers(1, Katan) Then
       Katan = Index2
     End If
```

```
  Next
  If Katan <> Index Then
    Temp1 = PhoneNumbers(1, Index)
    Temp2 = PhoneNumbers(2, Index)
    PhoneNumbers(1, Index) = PhoneNumbers(1, Katan)
    PhoneNumbers(2, Index) = PhoneNumbers(2, Katan)
    PhoneNumbers(1, Katan) = Temp1
    PhoneNumbers(2, Katan) = Temp2
   End If
Next
Debug.Print
Debug.Print "Sorted"
For Index = 1 To NumNames
  Debug.Print PhoneNumbers(1, Index), PhoneNumbers(2, Index)
Next
End Function
```

רשומות

עד עכשיו, עסקנו בנתונים מהסוגים הבאים:

- מספרים ממשיים.
- מספרים שלמים.
- ערכים בוליאניים.
- ערכים מסוג DateTime.
- ערכים מסוג מטבע.
- מחרוזות של אחד או יותר תווים.

בקבוצת נתונים שיש קשר לוגי ביניהם וכל נתון הוא מאותו סוג, נוכל להחזיק את הקבוצה במערך, ואם יש קבוצות רבות כאלו, אפשר להחזיקן במערך דו-מימדי. אבל, לעיתים יש צורך לטפל בקבוצת משתנים שמרכיביה הם מסוגים שונים. למשל, בדוגמה הקודמת, התייחסנו לשם ומספר טלפון. משום שהגדרנו את שניהם כסוג Variant, יכולנו להחזיק אותם באותו מערך. אבל אם נתייחס לשם כמשתנה מסוג מחרוזת, ולמספר טלפון כמשתנה נומרי, תהיה לנו קבוצת נתונים שמחזיקה נתונים קשורים מסוגים שונים. קבוצת נתונים כזו נקראת **רשומה** (Record), וכל איבר בה נקראת **שדה** (Field). השדות ברשומה אינם צריכים להיות מאותו סוג. קבוצת רשומות העוסקות בנושא מסוים מוחזקת כטבלה, כמו למשל קבוצת תלמידים.

מגדירים רשומה בחלק ההצהרות (Declaration) של המודול. להלן הצהרה עבור רשומת נתוני טלפון:

```
Type PhoneRecord
    Name As String
    Number As Long
End Type
```

שם הרשומה הוא PhoneRecord, ושמות השדות הם: Name ו-Number.

תוכנית C09:

```
Public Function C09()

Dim PhoneNumbers() As PhoneRecord
Dim EndOfData As Boolean
Dim Temp1, Temp2
Dim Index As Integer, Index2 As Integer, NumNames As Integer
Dim Katan As Integer
' Input data in telephone list
Do Until EndOfData
  Index = Index + 1
 '  Note Preserve only functions on the last dimension
  ReDim Preserve PhoneNumbers(1 To Index)
  PhoneNumbers(Index).Name = InputBox("הזן שם החבר")
  PhoneNumbers(Index).Number = InputBox("הזן מספר טלפון של החבר")
  If (MsgBox("האם סיימת?", vbQuestion + vbYesNoCancel) = vbYes) Then
    EndOfData = True
  Else
    EndOfData = False
  End If
Loop
NumNames = Index

' Print out in original order
Debug.Print
Debug.Print "Original Order"
For Index = 1 To NumNames
  Debug.Print PhoneNumbers(Index).Name
  Debug.Print PhoneNumbers(Index).Number
Next

' Sort names in the telephone list
For Index = 1 To NumNames - 1
  Katan = Index
  For Index2 = Index + 1 To NumNames
```

```
      If PhoneNumbers(Index2).Name < PhoneNumbers(Katan).Name Then
        Katan = Index2
      End If
   Next
   If Katan <> Index Then
     Temp1 = PhoneNumbers(Index).Name
     Temp2 = PhoneNumbers(Index).Number
     PhoneNumbers(Index) = PhoneNumbers(Katan)
     PhoneNumbers(Index) = PhoneNumbers(Katan)
     PhoneNumbers(Katan).Name = Temp1
     PhoneNumbers(Katan).Number = Temp2
   End If
Next
Debug.Print
Debug.Print "Sorted"
For Index = 1 To NumNames
   Debug.Print PhoneNumbers(Index).Name
  Debug.Print PhoneNumbers(Index).Number
Next
End Function
```

התוכנית דומה לתוכנית C08, אך נתוני מספרי הטלפון מוחזקים כרשומות במערך. אנו מצהירים בפקודת Dim שהמערך מחזיק איברים מסוג PhoneRecord, שהם רשומות.

שים לב שההתייחסות לכל שדה ברשומה נראית כך :

```
shem-maarach (index).sade
```

למשל,

```
PhoneNumbers(Index).Name
```

למעט שינוי זה, התוכנית זהה לתוכנית C08. הרבה יותר טבעי להחזיק את הנתונים כמערך רשומות מאשר כמערך דו-מימדי שאיבריו הם מסוג Variant, כדי שאפשר יהיה להחזיק באותו מערך שמות ומספרים.



תרגיל 16:

נניח שעבור כל פועל במפעל מסוים, קיימים הנתונים הבאים:

- שם
- משכורת
- אחד משלושת הציונים: א, ב, ג המגדירים את איכות עבודתו.

נתונים אלה אינם ממוינים. יש לקרוא אותם למחשב, לערוך את הפלט הנדרש, ולהציג אותו.

הפלט הוא רשימת עובדים ממוינת בסדר אלפביתי של שם העובד. נתוני עובד כוללים שם, משכורת וציון. במקום קוד הציון, יש להדפיס את כינויו: 'מצוין' במקום 'א', 'טוב' במקום 'ב', ו'ממוצע' במקום 'ג'.

תרגיל 17:

נתון וקטור של ערכים לא מסודרים. כתוב תוכנית כדי למצוא את החציון, המוגדר כערך האמצעי של הווקטור. שלבי הפתרון:

- מסדרים את N האיברים בסדר עולה.
- אם N הוא אי-זוגי, החציון הוא הערך שמספרו הסידורי הוא (N+1)/2.
- אם N הוא זוגי, החציון הוא ממוצע שני האיברים שמספריהם הסידוריים הם N/2 ו-N/2+1.

תרגיל 18:

קלוט מטריצה ובה הנתונים הבאים על עובדים:

- שם העובד,
- מספר האישי שלו,
- עיר מגורים,
- IQ.

הוצא את הנתונים האלה ממוינים לפי הסדרים הבאים:

- לפי שם העובד,
- לפי מספרו האישי.
- לפי IQ, מהגבוה לנמוך.

בנוסף, הוצא רשימה של שמות העובדים ומספרם האישי, מקובצת לפי עיר מגורים. שם העיר יופיע ככותרת, ומתחת לכותרת כל עיר יופיעו שם העובד ומספרו האישי ממוינים לפי שם.

פרק 4

פרוצדורות

מבוא

קטע הוא חלק לוגי של אלגוריתם לפתרון בעיה, שאפשר לתחום אותו כיחידה שלמה. באלגוריתמים שיש בהם חלקים שיש לבצע פעמים אחדות במהלך הפתרון, השימוש בשיטת הקטעים יכול לפשט את המבנה הלוגי של הפתרון ואת התכנות. בתוכניות שהודגמו עד עכשיו, הוצגה רק האפשרות לכתיבת קטעים בגוף התוכנית, בדרך כלל בצורת לולאה. יש לכך מספר חסרונות:

- אם התוכנית ארוכה, קשה יותר להבדיל בין המהלך העיקרי שלה לפרטים. כאשר נוקטים בשיטה זו, יש ללמוד את כל התוכנית, לפני שאפשר לטפל בחלקים שלה.
- כאשר יש צורך לבצע קטע תכנות פעמים אחדות, כותבים אותו מתחילתו עד סופו בכל מקום שהוא דרוש. כדי להתגבר על בעיות אלו, משתמשים בשיטות תכנות של **שגרות** (Subroutines) ו**פונקציות** (Functions). אפשר לכנות שגרות ופונקציות בשם הכולל **פרוצדורות** (Procedures), המתאר תוכנית לביצוע פעולה מצומצמת שכיחה, או פעולה אשר חוזרת פעמים אחדות בתוכנית העיקרית. בהמשך נלמד מתי ואיך להשתמש בפרוצדורות.

שגרות

תוכניות D01-D02:

```
Public Function D01()
' Read input data until the flag variable indicates end of data
' Requires defining a special flag variable
Const EndOfData = -1   ' Serves as a flag
Dim Flag As Integer
Do While Flag <> EndOfDa
   Call D02
   ' D02
```

```
 If (MsgBox("האם סיימת?", vbYesNoCancel) = vbYes) Then
   Flag = EndOfData
 Else
   Flag = 0
 End If
Loop
End Function

Public Sub D02()
Dim Rate As Single, Hours As Single, Salary As Single
Rate = InputBox("מה שיעור השכר לשעה?")
Hours = InputBox("כמה שעות עבדת?")
Salary = Hours * Rate
MsgBox " המשכורת שלך היא:" & Format(Salary, "$#.00")
End Sub
```

זוג זה של תוכנית (D01) ושיגרה (D02) מפריד בין שני חלקי תוכנית B12. שיגרה D02 מכילה את הקטע לחישוב משכורות, והתוכנית הראשית D01 מכילה את הלולאה המנהלת את התהליך. הפקודות שמרכיבות את השיגרה D02 הוצאו מ-D01, ומהוות שיגרה נפרדת. ב-D01, מתייחסים לפקודות אלו על ידי כתיבת ()Call D02 או D02.

תוכניות D03-D04:

```
Public Function D03()
' Read input data until the flag variable indicates end of data
' Requires defining a special flag variable
Const EndOfData = -1   ' Serves as a flag
Dim Flag As Integer
Do While Flag <> EndOfData
  Call D04(Flag)
'  D04 Flag        Alternative Representation
Loop
End Function

Public Sub D04(Flag As Integer)
' Read input data until the flag variable indicates end of  data
' Requires defining a special flag variable
Const EndOfData = -1   ' Serves as a flag
Dim Rate As Single, Hours As Single, Salary As Single
' Dim Flag As Integer
' Do While Flag <> EndOfData
  Rate = InputBox("מה שיעור השכר לשעה?")
  Hours = InputBox("כמה שעות עבדת?")
  Salary = Hours * Rate
  MsgBox " המשכורת שלך היא:" & Format(Salary, "$#.00")
```

```
  If (MsgBox("?האם סיימת", vbYesNoCancel) = vbYes) Then
    Flag = EndOfData
  Else
    Flag = 0
  End If
' Loop
End Sub
```

בזוג D01-D02, השארנו את השאלה לגבי המשך הביצוע בתוכנית הראשית, D01. בדוגמה זו העברנו שאלה זו לשיגרה. אבל כדי שהתוכנית הראשית תדע את התשובה לשאלה, אם יש המשך או לא, מעבירים את התשובה בעזרת פרמטר או ארגומנט שנקרא Flag. כאשר כותבים בתוכנית הראשית את D03 Call D04 (Flag), או רק D04 Flag, הערך של Flag מועבר מהשיגרה לתוכנית הראשית.

תוכניות D05-D06:

```
Public Function D05()
' takes an entire name like John Will Friedman and outputs
Friedman, J W

Dim strName As String
Dim FirstName As String
Dim MidName As String
Dim LastName As String
Dim Length As Integer
Dim Location As Integer
Location = 1
strName = InputBox("בבקשה להזין שם פרטי, אמצעי ומשפחה", "סידור השם", "")
strName = Trim(strName)
Length = Len(strName)
FirstName = Left$(strName, 1)
Location = InStr(strName, " ")
Call D06(strName, Location)   ' gets rid of all blanks
MidName = Mid(strName, Location, 1)
Location = InStr(Location + 1, strName, " ")
Call D06(strName, Location)
LastName = Right$(strName, Length - Location + 1)
MsgBox LastName & ", " & FirstName & " " & MidName
End Function
Public Sub D06(strName As String, Loc As Integer)

Dim Char As String
Char = " "
Do Until Char <> " " Or Loc > Len(strName)
  Loc = Loc + 1
```

```
  Char = Mid(strName, Loc, 1)
Loop
  If Loc > Len(strName) Then Loc = 0 'needed if there are
  ' alot of blanks at the end
End Sub
```

זוג זה מבוסס על תוכנית B06 שמסדרת שמות המתקבלים בצורת: Alvin Charles Smith, ומעבירה אותם לצורת: Smith, A C. השיגרה D06, שהוצאה מתוכנית B06, מבטלת רווחים עד לתו הבא שאינו רווח. את השיגרה הזו קראנו פעמיים מ-D05, ולכן היה כדאי להעביר את הקטע לשיגרה כדי לשפר קריאות ולחסוך בקוד.

תוכניות D06-D07:

```
Public Sub D06(strName As String, Loc As Integer)

Dim Char As String
Char = " "
Do Until Char <> " " Or Loc > Len(strName)
  Loc = Loc + 1
  Char = Mid(strName, Loc, 1)
Loop
  If Loc > Len(strName) Then Loc = 0 'needed if there are alot of blanks at the end
End Sub

Public Function D07()
' takes a sentence and determines how many words are in it

Dim strSentence As String
Dim FirstName As String
Dim MidName As String
Dim LastName As String
Dim NumWords As Integer
Dim Location As Integer Location = 1
strSentence = InputBox("בבקשה להזין משפט שלם", "מספר מילים", "")
 strSentence = Trim(strSentence) + " "
Do Until Location = 0
  Location = InStr(Location, strSentence, " ")
  NumWords = NumWords + 1
  Call D06(strSentence, Location)  ' get rid of any extra blanks
Loop

MsgBox "מספר המילים במשפט הוא: " & NumWords
End Function
```

זוג זה מבוסס על תוכנית B07 שסופרת את מספר המילים במשפט. השיגרה D06 הוצאה מתוכנית B07, והיא אותה שיגרה שהוצאנו מתוכנית B06, וזה מדגיש עוד יתרון של השימוש בשגרות, הרב-שימושיות של אותן שגרות.



תרגיל 1:

נקלוט נתוני עובד על מספר שעות העבודה לחודש, ונתוני עובדים על שיעור התשלום לשעה. הכן תוכנית שקולטת נתונים אלה, מדפיסה את השכר המגיע לכל עובד, ומוציאה כפלט את ממוצע מספר השעות שהעובד עבד בחודש, ואת ממוצע השכר ששולם לו. השתמש בתוכנית ראשית ומספר שגרות.

תרגיל 2:

שיעור המס על הכנסה של אדם נשוי עד 5,000 שקל הוא 20%, ועל כל הכנסה מעבר ל-5,000 שקל, הוא 40%. על הכנסה של אדם בודד המרוויח עד 2,000 שקל הוא 10%, ועל כל הכנסה מעבר ל-2,000 שקל הוא 60%. כתוב שתי שגרות, אחת לאדם נשוי והשנייה לאדם בודד. בתוכנית הראשית קלוט מצב משפחתי (בודד או נשוי), ורמת הכנסה. התוכנית הראשית אמורה לקלוט נתונים עבור מספר לא קבוע מראש של אנשים, ותיתן כתוצאה את מס ההכנסה שכל אחד חייב לשלם, וגם ממוצע ההכנסה ומס ההכנסה של כל העובדים שנקלטו.

תרגיל 3:

כתוב שיגרה לקריאת מחרוזת המציינת מספר בינארי והפיכתה למספר עשרוני.

רמז: מספר בינארי 100 שווה ל: $1*2^2+0*2^1+0*2^2$

ארגומנטים מסוגים שונים

יש שתי דרכים איך להעביר ארגומנטים: **לפי התייחסות** (By Reference), **ולפי ערך** (By Value).

משמעות הדרך הראשונה - לפי התייחסות - היא שהנתונים בתוכנית הראשית ובשיגרה הם אותם נתונים, ומה שמועבר לשיגרה אינו נתון שתופס מקום מיוחד בזיכרון, אלא מצביע לנתון המקורי. כל שינוי המבוצע על ידי השיגרה ישפיע על ערך המשתנה בתוכנית הראשית. זה הרצוי בדרך כלל, וזו גם שיטה מהירה לעבודה, משום שאין צורך להעתיק כמויות נתונים גדולות אלא רק מצביעים, שהם כתובות של נתונים. שיטה זו היא ברירת המחדל.

תוכניות D08-D09:

```
Public Function D08()
Dim dblVariable As Double
dblVariable = 8
Debug.Print "Before: "; dblVariable
' Call D09(dblVariable) alternative ways of calling the
subroutine D09 dblVariable
Debug.Print "After: "; dblVariable
End Function

Public Sub D09(dblNumber As Double)
dblNumber = dblNumber ^ (1 / 3)
End Sub
```

זוג זה, שמחשב את השורש הריבועי של מספר, מציג 8 לפני קריאת השיגרה ו-2 לאחר קריאתה.

משמעות הדרך השנייה - לפי ערך - היא שמעבירים עותק מהנתונים בתוכנית הראשית לתוכנית המשנית, וכל השינויים שיתבצעו יהיו על העותק בלבד. ערך המשתנה המיוחס לתוכנית הראשית לא ישתנה. שיטה זו איטית יותר וגם אינה מחזירה את הערך שלשמה קראנו את השיגרה. היתרון היחיד של שיטה זו הוא, שאם אין צורך בערך החדש של המשתנה בתוכנית הראשית, שיטה זו מגנה עליו בתוכנית הראשית ומשמרת את ערכו המקורי.

תוכניות D10-D11: שימוש במילה ByVal לפני הגדרת הפרמטר בשיגרה כדי לגרום למעבר הפרמטר לפי ערך.

```
Public Function D10()
Dim dblVariable As Double
dblVariable = 8
Debug.Print "Before: "; dblVariable
' Call D11(dblVariable) alternative ways of calling the subroutine
D11 dblVariable
Debug.Print "After: "; dblVariable
End Function

Public Sub D11(ByVal dblNumber As Double)
dblNumber = dblNumber ^ (1 / 3)
End Sub
```

התוכנית תדפיס את הערך 8 גם לפני וגם אחרי משום שערך המשתנה dblVariable אינו משתנה בתוכנית הראשית, גם לאחר ביצוע השיגרה. רק אם נבקש תדפיס מתוך השיגרה עצמה, נראה ערך שונה עבור הפרמטר.



תרגיל 4:

חזור על התרגילים הקודמים, ודייק לגבי סוג הפרמטרים, כך שרק פרמטרים שחייבים להשתנות יוכלו להשתנות.

ארגומנטים אופציונליים

ארגומנטים יכולים להיות גם אופציונליים, כלומר, אין הכרח לכלול את כולם בזמן הקריאה לשיגרה. כדי להפוך ארגומנט לאופציונלי, מוסיפים את המילה השמורה Optional לפני הגדרתו בכותרת השיגרה. אם ארגומנט כלשהו מוגדר כאופציונלי, כל הארגומנטים המוגדרים אחריו חייבים גם להיות מוגדרים כאופציונליים.

תוכניות D12-D13:

```
Public Function D12()
Dim datPaid As Date, datInvoiced As Date, strStatus As String
datInvoiced = #1/1/96#
datPaid = #3/2/96#
' A:  D13 strStatus, datInvoiced
' B: D13 datInv:=datInvoiced, strStat:=strStatus, datPay:=datPaid
C: D13 datInv:=datInvoiced, strStat:=strStatus
MsgBox strStatus & Chr(13) & " Date of Invoice: " & CStr(datInvoiced)
End Function
Public Sub D13(strStat As String, datInv As Date, Optional datPay)
If IsMissing(datPay) Then
  strStat = "Unpaid"
ElseIf datPay - datInv > 30 Then
  strStat = "Paid Late"
Else
  strStat = "OK"
End If
End Sub
```

כפי שנראה בשיגרה, הפרמטר DatPay הוא אופציונלי. כדי לדעת מה לעשות כאשר הוא אינו כלול בקריאה לשיגרה, משתמשים בפונקציה IsMissing. פונקציה זו יודעת לגלות אם ערך עבור אותו ארגומנט הגיע לשיגרה, ואם לא הגיע היא מאפשרת לנו לקבוע מה לעשות. במקרה שלנו, המשתנה strStat יקבל ערך של Unpaid. שים לב, שהסוג של datPay אינו מוגדר בשיגרה. היות וברירת המחדל עבור הסוג הוא Variant, זה כאילו הגדרנו אותו כ-Variant. הפונקציה IsMissing מקבלת ארגומנט מסוג Variant בלבד, ולכן הגדרנו אותו כך.

שורה A: קריאה לשיגרה D13. הארגומנט האחרון, שהוא אופציונלי, לא נכלל, וכתוצאה מזה הערך שיופיע ב: MsgBox יהיה: Unpaid.

שורה B: דרך אלטרנטיבית לקרוא לשיגרה. השתמשנו בארגומנטים בעלי שמות (Named Arguments). לפי שיטה זו, בזמן הקריאה לשיגרה רושמים את שם הארגומנט כפי שהוא מופיע בהגדרתו בשיגרה, אחר את הסימון =: ולבסוף את הערך המועבר לארגומנט, אם הוא משתנה או קבוע (בדוגמה שלנו הוא משתנה). יתרון שיטה זו הוא שאפשר לכתוב את הארגומנטים בזמן הקריאה לשיגרה בסדר כלשהו, ואם יש פרמטר או פרמטרים אופציונליים, לא כוללים אותם בקריאה לשיגרה. במקרה זה כללנו את כל הארגומנטים, והתשובה שווה ל: "Paid Late"

שורה C: כמו שורה B, רק שלא כללנו את הפרמטר האופציונלי, ולכן התוצאה היא כמו בשורה A.

תוכניות D14-D15: תוכנית ראשית ושיגרה, כאשר יש שני ארגומנטים הם אופציונליים.

```
Public Function D14()
Dim Par1 As Integer, Par2 As Integer, Par3 As Integer
Par3 = 30
' A: D15 Par1, , Par3
' B: D15 Param3:=37, Param1:=Par1, Param2:=77
C: D15 Param3:=37, Param1:=Par1
MsgBox Par1
End Function

Public Sub _
D15(Param1 As Integer, Optional Param2 As Integer, Optional Param3 As Integer)

If Param2 * Param3 > 300 Then
  Param1 = 300
Else
  Param1 = 100
End If
MsgBox "param2 = " & Param2
End Sub
```

שורה A: החסרנו את הארגומנט השני אך לא את השלישי. כדי להראות שהשני חסר, רושמים שני פסיקים, אחד ליד השני, במקום שהיה צריך להופיע הארגומנט השני. כאשר ארגומנט חסר ולא משתמשים בפונקציה IsMissing, הארגומנט שחסר מקבל את ברירת המחדל עבור אותו סוג. היות וסוג הארגומנט החסר פה הוא Integer, ערכו מתקבל כ-0, ולכן Par1 פה מקבל ערך 100.

שורה B: שימוש בארגומנטים בעלי שמות. היות והמכפלה של Param2 ו-Param3 היא מעל ל-300, Param1 יקבל את הערך 300.

שורה C: השמטנו פה ארגומנט אחד, אבל היות ואנו עובדים עם ארגומנטים בעלי שמות, לא כללנו את הארגומנט שלא רצינו לכלול. כשנרצה להעביר לשיגרה מספר לא ידוע מראש של ערכים נשתמש במילת העזר: ParamArray כדי לעזור לנו לבנות מערך דינמי. סוג המערך חייב להיות Variant. ParamArray חייב להיות תמיד הארגומנט האחרון, אך מותר לכלול לפניו כמה ארגומנטים שרוצים.

תוכניות D16-D17:

```
Public Function D16()
D17 "חיים", 13, 14, 15
End Function
Public Sub D17(strName As String, ParamArray intGrades() As Variant)
Dim var As Variant
Dim intI As Integer
Dim sngAvg As Single, intSum As Integer
A: Debug.Print strName
B: For Each var In intGrades
      Debug.Print var
  Next var
C: For intI = 0 To UBound(intGrades)
      Debug.Print intGrades(intI)
      intSum = intSum + intGrades(intI)
  Next
D: sngAvg = intSum / (UBound(intGrades) + 1)
E: Debug.Print sngAvg
End Sub
```

בפונקציה D16 קוראים לשיגרה D17, עם מחרוזת אחת (הארגומנט הראשון, "חיים"), ועוד 3 ארגומנטים, אך יכולנו לכלול כל מספר ארגומנטים רצוי.

שורה A: הדפסת הפרמטר הראשון, השם.

שורה B: גודל ParamArray אינו ידוע, ולכן נשתמש בפקודה For Each, שמתאימה למערך דינמי שגודלו אינו ידוע. For Each יכולה לעבוד עם משתנה Variant בלבד.

שורה C: במערך דינמי ניתן גם להשתמש בפקודת For רגילה, כאשר גודל המערך נקבע בעזרת הפונקציה: Ubound (IntGrades), כאשר Ubound מחזירה את מספר המצביע הגדול ביותר שבשימוש במערך IntGrades. (הפונקציה Lbound מחזירה את מספר המצביע הנמוך ביותר). שים לב שהערך הראשון ב-ParamArray מאוחסן באיבר מספר 0.

שורות D-E: חישוב הממוצע.

תוכניות D18-D19: דרך מקבילה לחישוב ממוצע של מספר ציונים לא ידוע.

```
 Public Function D18()
' Read input data until the flag variable indicates end of data
' Requires defining a special flag variable
Const EndOfData = -1   ' Serves as a flag
Dim intGrades() As Integer
ReDim intGrades(0)
' ReDim intGrades(0) As Integer  - this is equivalent to previos 2 lines
Dim Flag As Integer
```

```
Do While Flag <> EndOfData
  intGrades(UBound(intGrades)) = InputBox("הזן ציונים?")
  If (MsgBox("האם סיימת?", vbYesNoCancel) = vbYes) Then
    Flag = EndOfData
  Else
    Flag = 0
    ReDim Preserve intGrades(UBound(intGrades) + 1)
  End If
Loop
D19 "חיים", intGrades()
End Function

Public Sub D19(strName As String, intValues() As Integer)
Dim intVar As Variant
Dim var As Variant
Dim intI As Integer
Dim sngAvg As Single, intSum As Integer
A: Debug.Print strName
B: intVar = intValues()
C: For Each var In intVar
      Debug.Print var
  Next var
D: For intI = 0 To UBound(intVar)
      Debug.Print intVar(intI)
      intSum = intSum + intVar(intI)
  Next
E: sngAvg = intSum / (UBound(intVar) + 1)
F: Debug.Print sngAvg
G: intSum = 0
H: For intI = 0 To UBound(intValues)
      Debug.Print intValues(intI)
      intSum = intSum + intValues(intI)
  Next
I: sngAvg = intSum / (UBound(intValues) + 1)
J: Debug.Print sngAvg
End Sub
```

בדוגמה זו, בתוכנית הראשונה אנו קוראים את הציונים אחד אחד, ומזינים אותם לתוך מערך דינמי. בכל הוספת ציון, מגדילים את המערך הדינמי ב-1. מעבירים את המערך הדינמי מסוג Integer לשיגרה D19 כפרמטר.

שורות B-C : הפקודה For Each עובדת רק על משתנים מסוג Variant. כדי להפוך את המערך intValues למערך מסוג Variant, מותר להשוות אותו למשתנה מסוג Variant, בדוגמה intVar. המשתנה intVar הופך להיות מערך מסוג Variant באותו גודל של המערך intValues, ולכן אפשר להתייחס אליו כמערך מסוג Variant בפקודות C-E.

שורות G-J : טיפול ישיר במערך intValues כמערך מסוג שלם.



תרגיל 5:

כתוב תוכנית שמקבלת מספר ומדפיסה אותו. לידו, היא משרטטת קו המורכב מהסימן מקף (-), שאורכו יחסי לערך המספר (היסטוגרמה).

רמז: קלוט את הנתונים לתוך מערך דינמי. בסוף הקליטה, עבור לשיגרה כדי לגלות את המספר המקסימלי, ומצא את הגורם שיתרגם את המספר לערך בין 0 ל-50. הכפל כל מספר בגורם זה, והזן אותו כקלט לשיגרה שתדפיס את ההיסטוגרמה.

תרגיל 6:

כתוב שיגרה שמנתחת תוצאות מבחנים.

נתוני הקלט:

- תשובות כל סטודנט (מערך). התשובות הן מספרים מ-1 עד 4.
- מספר זהות של כל סטודנט.
- מפתח התשובות (מערך), שהוא אוסף התשובות הנכונות.

פלט השיגרה הוא ציון הסטודנט בבחינה, המחושב לפי מספר התשובות הנכונות מחולק במספר השאלות. חלק את השיגרה לשיגרה ראשית הקוראת לשגרות משניות. השיגרה אמורה לעבוד עבור מספר שאלות שאינו ידוע מראש, וגם מספר סטודנטים לא ידוע מראש.

תרגיל 7:

שכתב את הדוגמאות והתרגילים על חיפוש ומיון שהוצגו בפרק הקודם, כך שבכל מקרה השיגרה הראשית תחולק לכמה שגרות-משנה.

תרגיל 8:

מיזוג מוגדר עבור שני מערכים ממוינים אשר רוצים לאחדם למערך אחד שגם הוא יהיה ממוין. יש שתי דרכים לבצע מיזוג:

- קרא את שני המערכים לתוך המחשב, הפוך אותם למערך גדול אחד, ומיין את המערך הגדול.
- קרא את שני המערכים לתוך המחשב. עבור דרך האיברים המקבילים בכל מערך לפי הסדר, ושים את הקטן בכל זוג במערך החדש. כאשר כלים האיברים באחד מהמערכים, אפשר להעתיק את איברי המערך השני למערך הממוזג בלי בדיקות נוספות.

הכן שתי שגרות ראשיות, אחת לפי כל שיטה. לכל שיגרה ראשית, הכן את שגרות המשנה הדרושות.

טווח ההכרה של משתנים ופרוצדורות

טווח ההכרה (Scope) - באיזה תחום המשתנה מוכר. משתנה שנוצר בפרוצדורה נגיש רק באותה פרוצדורה, ונקרא משתנה **מקומי** (Local). ניתן להגדיר משתנה נוסף בפרוצדורה אחרת באותו שם.

תוכניות D20-D21:

```
Public Function D20()
Dim txtName As String
txtName = "Chaim Dov"
D21
MsgBox txtName, 0, "Main Procedure"
End Function

Public Sub D21()
Dim txtName As Double
txtName = 17
MsgBox txtName, 0, "Subroutine"
End Sub
```

הגדרנו משתנה בעל אותו שם בתוכנית הראשית ובשיגרה, למרות שתוכנם וסוגיהם שונים. בתוכנית הראשית, המשתנה מחזיק את ערכו המקורי למרות שהרצנו את השיגרה. אורך החיים של משתנה מקומי הוא כל עוד הפרוצדורה בה הוגדר מתפקדת עדיין. ביציאה מהפרוצדורה, המשתנה מאבד את ערכו וחוזר לברירת המחדל.

משתנה **מודולרי** (Modular) - משתנה שניתן לראותו מכל פרוצדורה במודול. משתנה זה נוצר כאשר מצהירים עליו בחלק ההצהרות (Declarations) של המודול, אך יש להקצות לו ערך רק בפרוצדורה. הוא מחזיק את ערכו עד סוף ביצוע היישום.

תוכניות D22-D23:

```
Public Function D22()
txtCompany = "Egged"
D23
MsgBox txtCompany, 0, "Main Procedure"
End Function

Public Sub D23()
MsgBox txtCompany, 0, "Subroutine"
txtCompany = "Osem"
End Sub
```

בשיגרה מדפיסים את תוכן המשתנה txtCompany שניתן בתוכנית D22, ובתוכנית הקוראת מדפיסים את תוכנו כפי שתוקן בשיגרה. רואים שיש גישה למשתנה מכל פרוצדורה במודול.

משתנה **סטטי** (Static) - משתנה מקומי ששומר את ערכו בין קריאה לקריאה. במקום להגדיר אותו בתוך הפרוצדורה בעזרת Dim, משתמשים במילת העזר Static.

תוכניות D24-D25:

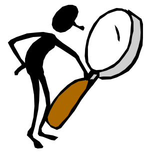
```
Public Function D24()
D25
D25
End Function

Public Sub D25()
' Dim intVar As Integer
Static intVar As Integer
intVar = intVar + 1
MsgBox intVar, 0, "Subroutine"
End Sub
```

כאשר מגדירים את המשתנה בעזרת Dim, הערך 1 מוצג פעמיים, אך כאשר מגדירים את המשתנה בעזרת Static, בפעם הראשונה 1 מוצג, ובפעם השנייה 2 מודפס. גם פרוצדורה יכולה להיות סטטית אם כותבים את המילה Static לפני שם הפרוצדורה. התוצאה מהוספת מילה זו היא שכל המשתנים בפרוצדורה יהיו סטטיים, אפילו אם הגדרנו אותם כרגיל ללא החלפת מילת ההצהרה Dim במילה Static.

משתנה **ציבורי** (Public) - משתנים הנגישים גם ממודולים אחרים, בנוסף למודול בו הוגדרו. משתנה ציבורי נוצר על ידי הצהרה בחלק ההצהרות (ולא בפרוצדורות), כאשר מחליפים Dim במילה Public. אבל אפשר להקצות לו ערך התחלתי רק בפרוצדורה, ולא בחלק ההצהרות. לכן, כדאי ליצור את כל המשתנים הציבוריים במודול אחד, ולהגדיר פרוצדורה באותו מודול בו יינתן ערך התחלתי לכל משתנה ציבורי. גם פרוצדורה יכולה להיות מוגדרת כציבורית, אם מוסיפים את המילה Public כמילה הראשונה בהגדרת הפרוצדורה. אם מוסיפים את המילה Private, הפרוצדורה תהיה נגישה רק לפרוצדורות באותו מודול בו הוגדרה. אם לא מגדירים לה סוג, ברירת המחדל היא שהפרוצדורה היא ציבורית.

גם **קבועים** (Constants) המוגדרים בעזרת Const בחלק ההצהרות יכולים להיות מלווים במילת העזר Private, ואז הם תקפים רק במודול בו הוצהרו, או Public (ברירת המחדל), ואז הם תקפים בכל המודולים בבסיס הנתונים.



הערה:

השתמש במשתנים ציבוריים רק כאשר זקוקים להם במודולים שונים. אפשר להוסיף קידומות לציון סוג המשתנה כדלהלן: m (מודולרי), g (גלובלי – ציבורי), ו- s (סטטי). משתנה מקומי יהיה ברירת המחדל, ולכן לא נקציב לו קידומת.

ההבדל בין פונקציה לשיגרה

ההבדל בין פונקציות ושגרות הוא בעיקר בדרך השימוש בהם.

תוכניות D26-D27:

```
Public Function D26()
Const intConst1 = 12
Const intConst2 = 24
Dim intTotFunc As Integer
intTotFunc = D27(intConst1, intConst2)
MsgBox intTotFunc, 0, "Sum of 2 Numbers using Function"
End Function

Public Function D27(int1 As Integer, int2 As Integer) As Integer
D27 = int1 + int2
End Function
```

פונקציה מחזירה ערך. כתוצאה מזה, היא חייבת להופיע בצד ימין של פקודת השמה, כאשר בצד השמאלי נמצא המשתנה שיחזיק את ערך הפונקציה.

היות והפונקציה עצמה מיוחסת לערך מסוים, יש לפונקציה סוג. בסוף כותרת הפונקציה D27, רואים את המילים As Integer, שמסמנות שהערך שייוחס לשם הפונקציה עצמה יהיה מסוג Integer. היות ויש ערך לשם הפונקציה, השם עצמו (D27) חייב להופיע בצד שמאל של פקודת השמה בתוך הפונקציה, כדי שהיא תקבל את הערך הרצוי. כמובן, לפעמים איננו רוצים לקבל החזר מפונקציה. במקרה כזה, לא חייבים לכתוב את סוג התוצאה בכותרת הפונקציה, וזה יהיה כאילו כתבנו As Variant, שזה ברירת המחדל. זה מסביר איך עבדנו עד עתה עם פונקציות, ואף פעם לא כתבנו את סוג התוצאה בכותרת. עד עכשיו השתמשנו בפונקציות כתוכניות ראשיות שלא היו צריכות להחזיר ערכים לתוכנית קוראת כלשהי.

תוכניות D28-D29: ביצוע אותו תפקיד בעזרת שיגרה.

```
Public Function D28()
Const intConst1 = 12
Const intConst2 = 24
Dim intTotSub As Integer
D29 intConst1, intConst2, intTotSub
MsgBox intTotSub, 0, "Sum of 2 Numbers using Subroutine"
End Function

Public Sub D29(int1 As Integer, int2 As Integer, intTot As Integer)
intTot = int1 + int2
End Sub
```

בתוכנית הראשית איננו צריכים פקודת השמה כדי לקלוט את התוצאות, אך אין ערך כלשהו המיוחס לשם השיגרה, ולכן אנו חייבים לקבל חזרה את התוצאה בעזרת ארגומנט נוסף בתוך השיגרה עצמה שקראנו לו intTotSub.



תרגיל 9:

כתוב פונקציה שמחשבת רווח ממכירת מוצרים. נתונים:

- מספר פריט (1 עד 10) - N,
- מחיר הפריט (P),
- תאריך (D),
- הכמות שנמכרה באותו תאריך (C),
- עלות שיווק משתנה (M) עבור כל יחידה, השונה עבור כל סוג של פריט.

השתמש בפונקציה זו בתוכנית שמדפיסה את הרווח ממכירת כל סוג בכל תאריך, וגם את הסכום הכולל של הרווחים בכל תאריך.

רמז: יש להבדיל בין שני סוגי קלט:

- קלט המאפיין כל סוג של פריט: N, P, M
- קלט המשתנה מיום ליום: N, C, D.



תרגיל 10:

Schum (sug as string, Netunim() as integer, n as integer) as Integer היא פונקציה שתכתוב לחישוב סכום ריבועי האיברים בשורה n או בעמודה n של המטריצה Netunim. אם "sug = "s מחשבים את סכום ריבועי האיברים בשורה n, ואם "sug = "a, מחשבים את סכום ריבועי האיברים בעמודה n. השתמש בפונקציה Schum בתוכנית שקוראת מטריצה של שלוש שורות ו-4 עמודות, ומחשבת את סכום הריבועים של כל שורה וכל עמודה. בפלט יש להציג את המטריצה המקורית, ובסוף כל שורה - את סכום הריבועים באותה שורה. בתחתית כל עמודה, יש לרשום את סכום הריבועים באותה עמודה.

תרגיל 11:

כתוב פונקציה שמחזירה את מספר הספרות במספר שהיא מקבלת כקלט.

תרגיל 12:

כתוב פונקציה שבודקת אם ספרה מסוימת כלולה במספר. השתמש בפונקציה זו בתוכנית שמגלה באיזה מספרי l בין 1 ל-100 נכללת הספרה d (בין 0 ל-9) ב-l, ב-l^2 וב-l^3.

פרק 5

שפת שאילתות סטנדרטית (SQL)

רשומות וטבלאות

כדי לטפל בקבוצת משתנים שמרכיביה הם מסוגים שונים, הגדרנו את המושג רשומה. למשל, אם מחזיקים נתוני עובד, אין זה נכון להניח שהרשימה כוללת שמות בלבד. סביר יותר להניח שיש ברשימה נתונים שונים עבור כל עובד: מספר עובד, שם, תאריך לידה, כתובת, מצב משפחתי, שכר או מרכיבי שכר וכדומה, אשר כל אחד מהם מוצג כסוג שונה. כפי שהוזכר קודם, קבוצת נתונים כזו נקראת **רשומה** (Record), וכל איבר בה נקרא **שדה** (Field). השדות ברשומה אינם חייבים להיות מאותו סוג נתונים. קבוצת רשומות העוסקות בנושא מסוים נקראת **טבלה** (Table). למשל, טבלת עובדים יכולה להחזיק את רשומות כל העובדים.

יצירת טבלאות

קיימת שפה מיוחדת לבניית טבלאות והפקת מידע מהם. שפה זו נקראת **Standard Query Language** (SQL).

המבנה הכללי של פקודת בניית טבלאות ב-SQL היא:

```
CREATE TABLE table name (field list)
```

כלומר, לאחר המילים CREATE TABLE כותבים את שם הטבלה הרצויה, ובתוך סוגריים את רשימת השדות שיהיו בטבלה, כולל סוג כל שדה, כאשר השדות מופרדים ביניהם בפסיקים.

תוכנית A1: בניית משפט SQL שיוצר טבלה.

```
Public Function A1()  'Create Table
Dim strSQL As String, booGood As Boolean, strQueryName As String, strTableName As String
1: On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query and table having this name, if they exist
2: strQueryName = "CreateCust"
3: strTableName = "Customers8"
4: DoCmd.DeleteObject acQuery, strQueryName
5: DoCmd.DeleteObject acTable, strTableName
' alternative way to erase a table using SQL rather than visual basic
' strSQL = "Drop Table " & strTableName
' SelectTable strSQL, strQueryName, booGood
' If booGood Then
'   DoCmd.OpenQuery strQueryName
' Else
'    MsgBox "Problem"
' End If

' Insert new table name and query name
6: strSQL = "Create Table " & strTableName & _
    "(IDCode Integer constraint IDCode PRIMARY KEY, LastName text (20) NOT NULL," & _
    "  FirstName text (20), StartDate DateTime NOT NULL, StreetNum text (20)," & _
    " CityState text (20), Zip text (10), Debt Currency)"


7: SelectTable strSQL, strQueryName, booGood
8: If booGood Then
     DoCmd.OpenQuery strQueryName
   Else
     MsgBox "Problem"
   End If
End Function
```

הסבר לפי שורות:

1: לפני שנבנה טבלה חדשה, יש לראות אם קיימת טבלה בעלת אותו שם, ולמחוק אותה (צעד 5). אך מה יקרה אם נבנה טבלה זו בפעם הראשונה. אם ננסה למחוק אותה, נקבל הודעת שגיאה. כדי לנטרל הודעה כזו, נודיע בשורה זו לא להתייחס אליהן.

2: מחרוזת המחזיקה את שם ה**שאילתה** (Query) שניצור כדי לבנות קובץ.

3: מחרוזת המחזיקה את שם הטבלה שניצור.

4-5: DoCmd נקרא אובייקט. אחרי הנקודה נרשום את שם ה**שיטה** (Method), שהיא: DeleteObject. שיטה זו מוחקת אובייקטים שונים, כאשר הארגומנט הראשון מציין את סוג האובייקט למחיקה (טבלה, שאילתה, טופס, דוח, מאקרו או מודול), והארגומנט השני מציין את שם האובייקט מהסוג שצוין כפרמטר הראשון.

בשורות ההערה שנציג, נשתמש בפקודה אחרת (ב-SQL) כדי למחוק את הטבלה. הפורמט עבור פקודה זו הוא:

```
DROP table name
```

בניית השאילתה להפעלת פקודה זו תוסבר בהמשך.

6: להלן הפקודה לבניית טבלה. מילים שמורות נכתבו באותיות רישיות:

```
CREATE TABLE  strTableName (IDCode Integer CONSTRAINT IDCode PRIMARY KEY,
LastName Text (20) NOT NULL,   FirstName TEXT (20),  StartDate DateTime NOT NULL,
StreetNum TEXT (20),  CityState TEXT (20), Zip TEXT (10), Debt Currency)
```

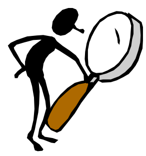
StrTableName מחזיק את שם הטבלה שנבנה. בסוגריים נרשום את השדות וסוגיהם.

IDcode – מספר הזהות. סוגו מוגדר כ-Integer. הביטוי Constraint מאפשר להגדיר **מפתח ראשי** (Primary Key) שגם שמו IDCode (אבל יכולנו גם לתת לו שם אחר).

LastName - משתנה מסוג טקסט (Text) שיכיל עד 20 תווים. הביטוי Not Null אוסר עלינו להשאיר שדה זה ריק (כי כך יקבל ערך NULL).

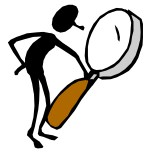
StartDate – תאריך הצטרפות הלקוח. נציין את סוגו כ-DateTime ונציין שחייבים למלא ערך זה.

Debt – משתנה מסוג מטבע (Currency).



הערות:

- פקודה זו אנו חייבים להגדיר כמחרוזת. היות וחלק מהמחרוזת מורכב ממשתנה, וגם אסור להמשיך מחרוזת אחת לשורה הבאה, אנו מפרקים אותה לחלקים, ומחברים ביניהם בעזרת &.
- במקום לתרגם את הפקודה למחרוזת, יכולנו ללחוץ על הכרטיסיה **שאילתות** ולכתוב אותה במתכונתה המקורית כשאילתה חדשה בתצוגת SQL. גישה זו לא טופלה פה.
- במקום לכתוב פקודה ב-SQL, יכולנו ללחוץ על הכרטיסיה **טבלאות** ולהגדיר את השדות בצורה אינטראקטיבית. גישה זו לא טופלה פה.
- במקום PRIMARY KEY, ניתן לכתוב את המילה UNIQUE, ואז יוגדר אינדקס ייחודי. על משמעות אינדקס ייחודי, וגם על כיצד לבנות אינדקס לא ייחודי, ראה להלן את תיאור פקודת CREATE INDEX.



כשיש מפתח ראשי כפול, ולא בודד, לא נגדיר אותו צמוד להגדרת שדה, שהרי אינו שייך לשדה כלשהו. במקום זאת, נגדיר אותו לאחר כל השדות, דוגמה: נניח שהמפתח הוא שם משפחה + שם פרטי. הגדרת הטבלה תיראה כך:

```
CREATE TABLE  strTableName (IDCode Integer, LastName Text (20)
NOT NULL,  FirstName TEXT (20),  StartDate DateTime NOT NULL,
StreetNum TEXT (20),  CityState TEXT (20), Zip TEXT (10), Debt
Currency), CONSTRAINT IDData PRIMARY KEY (LastName,FirstName)
```

7: קריאה לשיגרה SelectTable שמרכיבה שאילתה שניתנת להרצה.

8: פתיחת השאילתה אם הורכבה בלי בעיות.

תוכנית SelectTable: הכנת הגדרת שאילתה. אחרי שהשאילתה מוכנה, אפשר להריץ אותה על ידי פתיחתה.

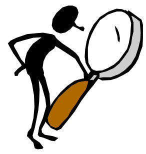
```
Public Sub SelectTable(strSequel As String, strQname As String, _
                    MakeQueryDef As Boolean)
1: Dim db As Database
2: Dim qdf As QueryDef
3: Set db = CurrentDb()
4: Set qdf = db.CreateQueryDef(strQname)
5: qdf.SQL = strSequel
41: 'Set qdf = db.CreateQueryDef(strQname, strSequel)
6: qdf.Close
7: MakeQueryDef = True
```

1-2: כפי שמגדירים משתנים מסוג נומרי וטקסטואלי, ניתן גם להגדיר משתנים המסמלים אובייקטים כמו בסיסי נתונים, טבלאות, טפסים, דוחות, וכולי. בשורות אלו הגדרנו משתנים לשני אובייקטים, **בסיס נתונים** (DataBase) ו**הגדרת שאילתה** (QueryDef – לא להתבלבל עם תוצאות השאילתה, שהן בעצם טבלה). אובייקטים נוספים הם TableDef (הגדרת טבלה), RecordSet (תוכן טבלה, או תוצאות שאילתה) ו-Field.

3: כדי לטעון ערך למשתנה מסוג אובייקט, משתמשים בפקודת Set, ולא בפקודת השמה פשוטה, כמו שעשינו עד כה עם משתנים רגילים. CurrentDb טוען את מסד הנתונים הנוכחי לתוך המשתנה בצד שמאל של המשוואה.

4: השיטה CreateQueryDef יוצרת הגדרת שאילתה ריקה, ונותנת לה את השם שמופיע בסוגריים (אם לא נרצה לשמור את השאילתה, נוכל להשמיט את הארגומנט ב-CreateQueryDef).

5: השמת המחרוזת שמחזיקה את SQL לשאילתה.

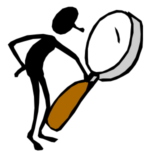


הערה:

אפשר לשלב את שורות 4 ו-5, כפי שעשינו בשורה 41. אפשר גם להשמיט את שם השאילתה ב-4 וב-41 ובמקומה לרשום מחרוזת ריקה (""), השאילתה תתבצע, אך לא תישמר. היא תיחשב כ**שאילתה זמנית** (Temporary Query).

6: סגירה ושמירת הגדרת השאילתה.

7: משתנה בוליאני זה הופך להיות True רק אם נגיע לסוף שיגרה זו. אם קיימת שגיאה בשיגרה, למשל אם פקודת SQL אינה תקינה, השיגרה לא תסתיים, המשתנה יישאר עם ערכו המקורי (False), ובתוכנית הראשית תופק הודעת שגיאה.



הערה:

את תוצאות התוכנית שלנו נוכל לראות בכמה מקומות במסד הנתונים. את השאילתה CreateCust נוכל לראות אם נבחר בכרטיסיה **שאילתות.** ניתן ללחוץ על **עיצוב** כדי לראות תצוגת עיצוב, ואז ללחוץ על הלחצן **תצוגה** שבסרגל הכלים כדי לראות **תצוגת SQL** וגם **תצוגת גיליון נתונים**. את הטבלה עצמה נוכל לראות אם נלחץ על הכרטיסיה **טבלאות.** ניתן ללחוץ על **עיצוב** כדי לראות תצוגת עיצוב, כלומר את מבנה הטבלה.

לאחר שראינו כיצד ליצור שאילתה עם פקודת SQL היוצרת טבלה, כדאי לציין שניתן להפעיל את פקודת SQL בעזרת השיטה Execute מבלי ליצור שאילתה פורמלית (תוכנית A1Alt). תוצאת ביצוע בצורה זו היא כמו תוצאת ביצוע שאילתה זמנית.



תרגיל 1:

הכן טבלת עובדים בעלת 3 שדות: שם (מפתח), שכר, וציון במבחן כניסה.

תרגיל 2:

בספריה מנהלים את מלאי הספרים. רשומת ספר כוללת את הפרטים הבאים:

- כותר
- מחבר
- שם המו"ל
- שנת הוצאה לאור
- מספר סידורי

הכותר והמחבר יחד מרכיבים את המפתח הראשי. בנה את הטבלה המתאימה.



תרגיל 3:

בבית ספר מחזיקים את הנתונים הבאים על הסטודנטים:

- שם פרטי
- שם משפחה
- מספר זהות
- מספר ילדים במשפחה
- מקצוע האב
- מקצוע האם
- שנות לימוד של האב
- שנות לימוד של האם
- ציון ממוצע בלימודים

המפתח הוא שם משפחה + שם פרטי. בנה את הטבלה המתאימה.

מילוי טבלאות

לאחר שיצרנו את מבנה הטבלה, נרצה למלא אותה בנתונים.

תוכנית ()Addrec:

```
Function Addrec()
' the new record will be placed in the proper place based on its
' key value. Won't allow you to put duplicate value for primary
' index. You can't add records to snapshots
Dim MyDb As Database, MyTable As Recordset
' Set mydb = DBEngine.Workspaces(0).Databases(0)
1: Set MyDb = CurrentDb()
Dim TableName As String
Const EndOfData = -1   ' Serves as a flag
Dim Flag As Integer
2: TableName = "Customers8"
3: Set MyTable = MyDb.OpenRecordset(TableName, dbOpenTable)
' instead of update we use add, since we are adding a
' blank record (similar to insert)
4: Do While Flag <> EndOfData
   MyTable.AddNew
   MyTable![LastName] = InputBox("מה שם המשפחה?")
   MyTable![FirstName] = InputBox("מה השם הפרטי?")
   MyTable![IDCode] = InputBox("מה מספר הזהות?")
   MyTable![StartDate] = InputBox("מה תאריך ההתחלה?")
   MyTable![CityState] = InputBox("מה עיר ומדינה?")
```

```
    MyTable![StreetNum] = InputBox("מה הכתובת")
    MyTable![Zip] = InputBox("?מה המיקוד")
    MyTable![Debt] = InputBox("?מה גודל החוב")
    MyTable.UPDATE ' update values just added to the blank record
    If (MsgBox("?האם סיימת", vbYesNoCancel) = vbYes) Then
      Flag = EndOfData
    Else
      Flag = 0
    End If
    Loop
5: MyTable.Close
End Function
```

הסבר: MyTable מוגדר כמשתנה המחזיק אובייקט מסוג RecordSet, כשהכוונה היא לאוסף רשומות.

3: הפקודה העיקרית למילוי משתנה מסוג RecordSet נראית כך:

```
SET recordset = object.OPENRECORDSET (source, type, options, lockedits)
```

object יכול להיות מסד נתונים, ואז נגדיר את **מקור אוסף הרשומות** (Source Recordset) כטבלה או שאילתה. הוא יכול להיות גם הגדרת טבלה, הגדרת שאילתה, או Recordset אחר, ואז אין צורך להגדיר אותו, שהרי הוא מובן מאליו.

4: לולאה בה מופעלות על הטבלה MyTable השיטות AddNew כדי להוסיף רשומות חדשות, ואחר השיטה Update כדי לעדכן. בלולאה ממלאים את השדות אחד אחד.

Type - סוג Recordset. קיימים 4 סוגים:

- **טבלה** (Table) - מסומן על ידי הקבוע dbOpenTable. רק הרשומה השוטפת מוחזקת בזיכרון. לפני השימוש קובעים לפי איזה אינדקס (סדר) יוחזרו הנתונים. מיועד לטיפול בטבלה עצמה, וכתוצאה מזה אפשר לנצל את המפתחות והאינדקסים שהוגדרו עבור הטבלה, להוספה, מחיקה ועדכון רשומות. משום השימוש באינדקסים סוג זה **יעיל יותר בחיפושים ומיונים**. זהו סוג ברירת המחדל אם הפרמטר אינו כלול. שלא כמו קבוצות דינמיות ותצלומי בזק, סוג זה אינו יכול להתייחס ליותר מטבלה אחת או לתוצאות שאילתה.
- **קבוצה דינמית** (Dynaset) - קבוצת רשומות שיכולה להכיל רשומות המורכבות משדות מטבלה אחת או יותר (כלומר תוצאות שאילתה). סוג זה שומר את המצביע לרשומה, ולא את הרשומה עצמה. בכל עדכון או תצוגה ש-Dynaset מבצעת, היא קוראת לכל הרשומה. היא אינה חלק ממסד הנתונים, אך היא משקפת את השינויים שבו, גם הטבלאות במסד הנתונים משקפות שינויים שבוצעו על הקבוצה הדינמית. אין משמעות למפתחות או אינדקסים שהוגדרו על טבלאות שהן מקור השאילתה, ולכן הן אינן מנצלות אינדקסים. קבוצה דינמית מסומנת על ידי הקבוע dbOpenDynaset. סדר הרשומות הוא לפי הביטוי Order By בשאילתה, או לפי ברירת המחדל של סדר הרשומות שבתוצאת השאילתה.

- **תצלום בזק** (Snapshot) - קבוצה סטטית של רשומות לבדיקת נתונים בטבלה או תוצאת שאילתה. Snapshot יכול להכיל שדות מאחת או יותר טבלאות במסד הנתונים, אבל אינו מסוגל לעדכן אותן, משום שהוא רק עותק של המקורי, לא קבוצת קשרים ומצביעים כמו הסוגים טבלאות או קבוצות דינמיות. תוכן כל הנתונים מועבר לזיכרון, חוץ משדות OLE ותזכיר (Memo). לאחר הטעינה, הוא אינו משקף שינויים בטבלאות שעליהן הוא מבוסס. מסומן על ידי הקבוע dbOpenSnapshot.
- **קדימה לבד** (Forward-Only) - זהה לתצלומי בזק, אך יכול לנוע קדימה בלבד. כשיש צורך במעבר קדימה בלבד הוא משפר ביצועים. מסומן על ידי הקבוע dbForwardOnly.

סיכום: לקבוצות נתונים קטנות, תצלומי בזק יותר טובים לפתיחה וגלילה, זאת בתנאי שאין צורך בעדכון. אם מעבר אחד מספיק, קדימה לבד הוא המתאים ביותר. לקבוצות גדולות יותר קבוצות דינמיות טובות יותר, משום שהן אינן מחייבות הורדת כל הרשומות, רק את שדות הקשר. אם הנתונים נלקחים מטבלה, הסוג טבלה מהיר יותר, משום שאפשר להשתמש באינדקסים שלה. מספר גדול של טבלאות מקור ועמודות משפיע לרעה בעיקר על קבוצות דינמיות.

תוכנית ()Addrec2:

```
Function Addrec2()
' the new record will be placed in the proper place based on its
' key value. Won't allow you to put duplicate value for primary
' index. You can't add records to snapshots
Dim MyDb As Database, MyTable As Recordset
' Set mydb = DBEngine.Workspaces(0).Databases(0)
Set MyDb = CurrentDb()
Dim TableName As String
Const EndOfData = -1   ' Serves as a flag
Dim Flag As Integer
TableName = "Customers8"
Set MyTable = MyDb.OpenRecordset(TableName, DB_OPEN_TABLE)
' instead of update we use add, since we are adding a
' blank record (similar to insert)
With MyTable
Do While Flag <> EndOfData
   .AddNew
   ![LastName] = InputBox("מה שם המשפחה?")
   ![FirstName] = InputBox("מה השם הפרטי?")
   ![IDCode] = InputBox("מה מספר הזהות?")
   ![StartDate] = InputBox("מה תאריך ההתחלה?")
   ![CityState] = InputBox("מה עיר ומדינה?")
   ![StreetNum] = InputBox("מה הכתובת")
   ![Zip] = InputBox("מה המיקוד?")
   ![Debt] = InputBox("מה גודל החוב?")
```

```
   .UPDATE
   If (MsgBox("האם סיימת?", vbYesNoCancel) = vbYes) Then
      Flag = EndOfData
   Else
      Flag = 0
   End If
Loop
.Close
End With
End Function
```

פונקציה זו דומה לפונקציה Addrec, למעט הפקודה: With MyTable…End With שנוספה. פקודה זו מאפשרת לנו לפשט את הקוד על ידי הורדת הקידומות MyTable בכל פקודה שבין With ו- End With.



תרגיל 4:

מלא 10 רשומות בנתונים בדויים בטבלה **עובדים** שהוגדרה בקבוצת התרגילים הקודמת. בדוק שהנתונים הוזנו בהצלחה על ידי פתיחת הטבלה. בחלון מסד הנתונים בחר בכרטיסיה **טבלאות**, לחץ לחיצה כפולה על שם הטבלה שלך, או לחץ עליה ועל הלחצן **פתיחה**.

תרגיל 5:

מלא 10 רשומות בנתונים בדויים בטבלה שבנית עבור הספריה.

תרגיל 6:

מלא 10 רשומות בטבלה שבנית עבור בית ספר בנתונים האלה:

פרטי	משפחה	ילדים	שנות לימוד אב	שנות לימוד אם	ציון ממוצע
אברהם	פדר	7	7	7	81
חנוך	כהן	2	15	14	94
שרה	פרידמן	5	6	8	65
שלום	עדידה	4	10	11	88
חוה	כהן	3	14	12	61
יהודית	פדר	1	16	16	98
שלום	עמיר	2	18	14	85
אברהם	מזרחי	9	8	13	92
יעקב	הנובר	2	9	7	68
דניאל	לוי	6	7	8	82
דניאלה	רובין	4	10	9	77
אסתר	הורביץ	5	13	14	92

עריכה

תוכנית ()Alter: שאילתה המאפשרת הוספה והורדת שדות מטבלה.

```
Public Function Alter()
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query and table having this name, if they exist
strQueryName = "AlterCust"
strTableName = "Customers1"
DoCmd.DeleteObject acQuery, strQueryName
' Add a new field. Use Drop Column to Delete a column
strSQL = "Alter Table " & strTableName & " ADD COLUMN OldDebts Currency;"
'OldDebts are > 1 year old
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

שאילתה זו משתמשת בפקודה:

```
ALTER TABLE  table name  ADD COLUMN column name column description
```

ב-SQL קיימת פקודה מקבילה למחיקת שדות:

```
ALTER TABLE  table name  DROP COLUMN column name
```

תוכנית ()Alter2: שיטת Execute להוספת שדות, במקום בניית שאילתה פורמלית.

```
Public Function Alter2()
Dim strSQL As String, strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous table having this name, if they exist
Dim db As Database
Set db = CurrentDb()
strTableName = "Customers2"
' OldDebts are > 1 year old
strSQL = "Alter Table " & strTableName & " ADD COLUMN OldDebts Currency;"
db.Execute strSQL
End Function
```

לאחר הוספת שדה, נרצה למלא אותו בנתונים.

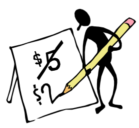
תוכנית ()Fillrec:

```
Function Fillrec()
' The new field will be filled
Dim MyDb As Database, MyTable As Recordset
' Set mydb = DBEngine.Workspaces(0).Databases(0)
Set MyDb = CurrentDb()
Dim TableName As String
TableName = "Customers1"
Set MyTable = MyDb.OpenRecordset(TableName, dbOpenTable)
' we use update to change existing records
With MyTable
Do Until MyTable.EOF
1:    .Edit
2:    ![OldDebts] = InputBox("כמה החוב עבור " & ![IDCode])
3:    .UPDATE
4:    .MoveNext
Loop
.Close
End With
End Function
```

בשיגרה זו מסומנים 4 שלבים בעדכון טבלה :

1. פתיחת קבוצת הרשומות לעריכה.
2. תיקון הערכים שעומדים לשינוי.
3. עדכון הטבלה.
4. מעבר לרשומה הבאה (לפי האינדקס הקיים כעת).

שים לב שביצוע הפקודה Do Until נמשך עד ל- MyTable.EOF שמשמעותו היא סוף הרשומות בטבלה המכונה MyTable (EOF - ראשי תיבות עבור End Of File).



תרגיל 7:

הוסף לטבלת עובדים שבנית שדות שייקראו **כתובת עיר** ו**מין**.

תרגיל 8:

הוסף לטבלת הספריה שדה שיכיל את מספר ISDN של כל ספר.

תרגיל 9:

הסר מטבלת בית הספר את השדות **מקצוע האב ומקצוע האם**. הוסף את השדה **כתובת**.

שאילתות פשוטות

שאילתות מאפשרות לנו לבחור עמודות ו/או שורות מסוימות מטבלאות בודדות ומשולבות, ולהציג אותן יחד. את ההגדרות שתקבענה אילו שורות ועמודות להציג, נכתוב בשפת SQL.

המבנה הכללי של SQL להגדרת שאילתה הוא:

```
SELECT expression FROM  tables [WHERE condition] [ORDER BY expression]
```

תוכנית A2: שימוש במבנה הבסיסי עבור Select

```
Public Function A2()   'Basic Select Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SelectCust"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers1"
' Using AS to define an Alias, where [] are needed since a space is included
1:  strSQL = "Select IDCode, LastName From " & strTableName
2: ' strSQL = "Select IDCode, LastName as [Family Name] From " & strTableName
3: ' strSQL = "Select IDCode, LastName, Debt + OldDebts as Balance From " &
strTableName
4: ' strSQL = "Select IDCode, LastName, CityState From " & strTableName & _
   '          " Order By CityState"
5: '          " Order By CityState Desc, LastName"

6:
    '        " Where CityState = 'תל אביב'"
7:
 '     " Where CityState = 'תל אביב' OR CityState = 'טבעון'"
8:
 '       " Where CityState In ('תל אביב','טבעון')"
9:
 '       " Where CityState In ('תל אביב','טבעון') AND IDCode > 59"
10:
' " Where CityState In ('תל אביב','טבעון') and IDCode > Num"
 SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

1: בחרנו מהטבלה שתי עמודות בלבד (IDCode ו-LastName) ללא תנאים נוספים.

2: כמו ב-1 בהוספת המילה AS, שגורמת לכך שהתוצאה תוצג בעמודה עם הכותרת Family Name ולא עם הכותרת LastName.

3: השדה המחושב (Debt + OldDebts) מופיע כמו כל שדה אחר, אך הוא כולל את השדות והאופרטורים הנחוצים להגדרתו.

4: כמו ב-1, בהוספת הביטוי ORDER BY, שמסדר את התוצאה לפי שדה CityState. ברירת המחדל היא סדר עולה.

5: כמו 4, אך המיון הוא לפי שני מפתחות. הראשון הוא CityState בסדר יורד (DESC). כשיש מספר רשומות שלהן ערך זהה ב-CityState, המיון יתבצע לפי LastName בסדר עולה (ברירת המחדל).

6: דוגמה לביטוי WHERE (בהמשך לפקודת SQL בשורה 4). הקפנו את המחרוזת הפנימית בגרשים בודדים.

7: WHERE עם תנאי מורכב המשתמש באופרנד OR לקשר בין התנאים.

8: WHERE עם תנאי המשתמש באופרנד IN המחייב בחירת אחד מהערכים המופיעים בסוגריים.

9: WHERE עם תנאי מורכב המשתמש באופרנד AND לקשר בין התנאים.

10: כמו 9, אלא שבמקום קבוע, כתבנו את המשתנה NUM. לפני הרצת השאילתה, Access תבקש להזין ערך לפרמטר זה.

לפעמים איננו יודעים מראש את ערך התנאי שברצוננו לכפות. אפשר לקלוט ערך, ורק אז לשבץ את המשתנה המחזיק את ערכו לביטוי WHERE. אפשר אף לקלוט את שם השדה שעליו מחילים את התנאי, ולאפשר למשתמש בחירה בשדה שעליו נפעיל תנאי.

תוכנית A2Input:

```
Public Function A2Input()   'Basic Select Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
   strTableName As String, strCity As String, intID As Integer, strField As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                     ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SelectCust"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers1"
' Example 1 - takes literal input
' strCity = Trim(InputBox("Print City Name in Hebrew"))

' strSQL = "Select IDCode, LastName From " & strTableName & _
'       " Where CityState = '" & strCity & "'"
```

```
' Example 2 - takes numeric input
' intID = InputBox("Print minimum ID")

' strSQL = "Select IDCode, LastName From " & strTableName & _
'          " Where IDCode > " & CStr(intID)

' Example 3 - allows inputting field as well as value
strField = InputBox("What Field?")
intID = InputBox("Print ID")
 strSQL = "Select IDCode, LastName From " & strTableName & _
          " Where " & strField & " = " & CStr(intID)

 SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

Example 1 - נקלוט את שם העיר בה מתגוררים הלקוחות הרצוים.

Example 2 - נקלוט את מספר הזהות המינימלי של הלקוחות הרצוים.

Example 3 - נקלוט את שם השדה שעליו ברצוננו להחיל את התנאי, ואת ערך התנאי.

מי שמעוניין במידע נוסף על הגדרות שאילתות (QueryDefs) יכול לעיין בעזרה בערך SQL, ולבדוק Parameter Object ו-Filter Property.

תוכנית A2Query:

```
Public Function A2Query()   'Basic Select Query as report
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String, db As Database, rst As Recordset, intCustomers As Integer
On Error Resume Next 'needed if the query we wish to delete doesn't
                     ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SelectCust"
DoCmd.DeleteObject acQuery, strQueryName
' strTableName = "Customers1"
Set db = CurrentDb
strSQL = "Select * From Customers where City = 'ירושלים'"
Set rst = db.OpenRecordset(strSQL, dbOpenSnapshot)
rst.MoveLast
intCustomers = rst.RecordCount
```

```
MsgBox intCustomers
SelectTable strSQL, strQueryName, booGood
Loop1:
If booGood Then
      DoCmd.OpenQuery strQueryName
Else
      MsgBox "Problem"
End If

If MsgBox("Are you Finished?", vbQuestion + vbYesNoCancel) = vbYes Then
    DoCmd.Close
Else
   GoTo Loop1
End If

strQueryName = "SelectCust"
DoCmd.DeleteObject acQuery, strQueryName
strSQL = "Select * From Customers"
Set rst = db.OpenRecordset(strSQL, dbOpenSnapshot)
rst.MoveLast
intCustomers = rst.RecordCount
MsgBox intCustomers
SelectTable strSQL, strQueryName, booGood
If booGood Then
      DoCmd.OpenQuery strQueryName
Else
      MsgBox "Problem"
End If
If MsgBox("Are you Finished?", vbQuestion + vbYesNoCancel) = vbYes Then
    DoCmd.Close
End If
End Function
```

נפתח את השאילתה כקבוצת רשומות, נשתמש בתכונה RecordCount כדי לברר את מספר הרשומות הכלולות בשאילתה, נציג את השאילתה ונסגור אותה. אחר נפתח את כל הטבלה כקבוצת רשומות, נשתמש שוב ב-RecordCount כדי לברר את מספר הרשומות בטבלה כולה ונציג את הטבלה.



תרגיל 10:

הצג רשימת עובדים (ראה תרגיל 1) מסודרת לפי ציון מבחן כניסה.

תרגיל 11:

הצג את השם והציון בלבד ברשימה מהשאלה הקודמת.



תרגיל 12:

הצג רשימת עובדים המשתכרים מעל 5,000 ש"ח לחודש.

תרגיל 13:

הצג את רשימת הספרים בספריה הכוללת כותר ומחבר. סדר הרשימה יהיה לפי ISDN (אך השדה עצמו לא יופיע ברשימה).

תרגיל 14:

קיימת השקפה שטוענת שציון במבחן הכניסה הוא פונקציה של שנות לימוד ההורים. על בסיס טבלת בית הספר, הוצא כפלט את סכום שנות הלימוד של ההורים וציון במבחן כניסה.

תרגיל 15:

אילו סטודנטים בטבלת בית הספר באים ממשפחות שבהן מעל 4 ילדים ובעלי ציון שהוא מעל 80.

תרגיל 16:

הקריטריון למילגה הוא סטודנט שבא ממשפחה שבה מעל 4 ילדים וציון מעל 90, מי יקבל מילגה?

שאילתות בטבלאות קשורות

כששתי טבלאות קשורות ניתן להציג שדות משתיהן. נציין את מהות הקשר בעזרת ביטוי שנוסיף לפקודת SELECT. הביטוי נקרא JOIN ולו שני סוגים:

- **INNER** (קשר פנימי - ברירת המחדל)
- ו-**OUTER** (קשר חיצוני), כאשר OUTER יכול להיות RIGHT או LEFT.

משמעות קשר פנימי היא קישור בין כל שתי רשומות שיש להן ערך שווה בשדה הקשר שלהם. למשל, אם מספר לקוח הוא 123 ויש 3 הזמנות עבור לקוח 123, קיים קישור בין לקוח 123 וכל אחת מ-3 רשומות אלו. מבנה ביטוי קשר פנימי שנוסף ל-SELECT הוא כדלהלן:

```
table1 INNER JOIN table2 ON field1 = field2
```

תוכנית A3: דוגמת קשר פנימי בין טבלת Customers וטבלת Orders.

```
Public Function A3()  'Join Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "JoinCustOrder1"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
' if the key in both is based on 2 fields, use AND, e.g.
' if the key were first name and last name, the link phrase would be:
' Customers.FirstName = Orders.FirstName AND
' Customers.LastName = Orders.LastName
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
' Using Join to connect 2 tables
'  Table name prefix needed only if 2 tables have field with same name
strSQL = "Select OrderID, OrderDate, Orders.CustomerID, CustomerLastName From " _
        & strTableName1 & " Inner Join " & strTableName2 & " ON " & LinkPhrase _
        & " Order BY OrderID "
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

תוכנית A3Alt:

```
Public Function A3Alt()  'Alternative Join Query without Join
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "JoinCustOrder2"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
```

```
' Connecting 2 tables by writing both and having a linkphrase
' Table name prefix needed only if 2 tables have field with same name
strSQL = "Select OrderID, OrderDate, Orders.CustomerID, CustomerLastName From " _
        & strTableName1 & ", " & strTableName2 & " Where " & LinkPhrase _
        & " Order BY OrderID "
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

אפשר גם לקשר בין שתי טבלאות על ידי רישום שתי טבלאות לאחר המילה FROM בפקודת SELECT, ואחר רישום ביטוי הקשר (field1 = field2) אחר המילה WHERE.

אפשר לבצע קשר כפול. למשל, טבלת לקוח קשורה לטבלת הזמנות וטבלת הזמנות קשורה לטבלת שורה בהזמנה. נניח שברצוננו לראות את השם שבכל שורת הזמנה. הקשר הכפול ייראה כך (תוכנית A4):

```
(table1 INNER JOIN table2 ON table1.field1 =table2. field2)  INNER JOIN table3 ON
table1.field1 = table3.field3
```

או כך (תוכנית A41):

```
table1 INNER JOIN (table2 INNER JOIN table3 ON table2.field2 =table3. Field3) ON
table1.field1 = table2.field2
```

שים לב ששתי הצורות הן עדיין מקבילות לצורה המקורית:

```
table1 INNER JOIN table2 ON field1 = field2
```

בצורה הראשונה table1 הוחלף על ידי הביטוי בסוגריים (שהוא בעצמו טבלה שנוצרה כתוצאה מקשר פנימי), ובצורה השנייה table2 הוחלף על ידי הביטוי בסוגריים. חשוב לזכור שתוצאת קשר פנימי היא בעצמה טבלה, ולכן היא יכולה להחליף את הטבלה שבנוסחה המקורית.

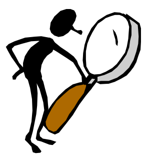
עד עתה הנחנו ששאילתה על טבלאות קשורות תציג כל רשומה בטבלה הראשונה שקשורה לרשומה מסוימת בטבלה השנייה. קיים גם מושג של קשר ימני (שמאלי), כאשר הכוונה שבתוצאה מוצגות כל הרשומות מהטבלה שמופיעה בצד הימני (שמאלי) בביטוי JOIN, ורק הרשומות הקשורות מהטבלה שמופיעה בצד השמאלי (הימני). הצורה הכללית של הביטוי JOIN עבור קשר ימני או שמאלי היא:

```
 table1 RIGHT JOIN table2 ON field1 = field2
  table1 LEFT JOIN table2 ON field1 = field2
```

תוכנית A5: דוגמת קשר שמאלי.

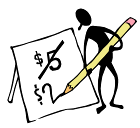
```
Public Function A5()  'Left and Right Join Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "LeftJoin"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
' if the key in both is based on 2 fields, use AND, e.g.
' if the key were first name and last name, the link phrase would be:
' Customers.FirstName = Orders.FirstName AND
' Customers.LastName = Orders.LastName
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
' Using Left Join to connect 2 tables
' Table name prefix needed only if 2 tables have field with same name
strSQL = "Select OrderID, OrderDate, Orders.CustomerID, CustomerLastName From " _
        & strTableName1 & " Left Join " & strTableName2 & " ON " & LinkPhrase _
        & " Order BY CustomerLastName"
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```



הערה:

כדי לבצע את התרגילים הבאים נוסיף טבלת קשר המקשרת בין טבלת בית הספר וטבלת הספריה. בטבלה יופיעו 3 שדות: מספר זהות של סטודנט, מספר סידורי של ספר, תאריך להחזרת הספר ושדה בוליאני שערכו TRUE כאשר ספר מוחזר. מטרת הטבלה היא לציין אילו ספרים הושאלו על ידי אילו סטודנטים. שני השדות הראשונים הם שדות מפתח. נקרא לטבלה **טבלת השאלות**. מלא את הנתונים על בסיס הנתונים הקיימים בשתי הטבלאות שעליהן מבוססת **טבלת השאלות**.



תרגיל 17:

הוצא כפלט את שמות הסטודנטים ששאלו ספרים מהספריה ואת מספרי הספרים ששאלו.



תרגיל 18:

הוצא כפלט את שמות הסטודנטים ששאלו ספרים מהספריה ואת שמות הספרים ששאלו.

תרגיל 19:

הוצא כפלט שם כל סטודנט ושם הספרים ששאל. גם אם לא שאל ספרים, שמו עדיין יופיע פעם אחת.

שאילתות סיכום

בנוסף ליכולת שאילתה להציג עמודות ושורות מטבלה אחת או יותר, היא גם יכולה לחשב סכום או ממוצע של כל הנתונים בעמודות הטבלה. במקום לרשום בפקודת Select שם של כל שדה רצוי להצגה נכתוב: function ([field name]). כתוצאה נקבל שורה אחת ובה תוצאות הפעלת הפונקציות על העמודות. הפונקציות האפשריות הן:

- Var
- Min
- Avg
- StDev
- Count
- Sum
- Max

תוכנית A6: שאילתת סיכום שמשתמשת במרבית הפונקציות.

```
Public Function A6()   'Basic Summary Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
   strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                  ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SummaryOrders"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Orders"
' Using AS to define an Alias, where [] are needed since a space is included
strSQL = "Select Sum(SubTotal) as [Sum of Subtotals], Avg(Shipping) as _
          [Ave Shipping], " & " Count (Orders.OrderID) as [Num Orders],
          Min(IncomeTax) as [Min Tax], Max(IncomeTax) as [Max Tax]" & " From " _
          & strTableName ' & " Where OrderID > 100"

SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

נניח שאין ברצוננו להציג סכום מכירות בכל העמודה, במקום זאת ברצוננו להציג את הסכום עבור כל ערך שונה בשדה כלשהו. למשל, נניח שברצוננו להציג את סכום המכירות בכל עיר (ולא את הסכום הכולל של המכירות). כדי לאפשר הקבצה נוסיף לפקודת Select את הביטוי Group By.

תוכנית A7: הקבצה לפי City **וסידור לפי הביטוי הראשון בפקודת** Select.

```
Public Function A7()   'Basic Summary Query with Grouping
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "GroupedOrders"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Orders"
' Using AS to define an Alias, where [] are needed since a space is included
strSQL = "Select Sum(SubTotal) as SumSub, Avg(Shipping) as AveShip, " _
& " Count (Orders.OrderID) as [Num Orders], Min(IncomeTax) as MinTax, " _
& "Max(IncomeTax) as [Max Tax], City" & " From " & strTableName _
& " Group By City Order By 1 desc"
        ' 1 causes it to be in descending order of the
        ' first expression here which is subtotal.
        ' Without this it is ordered by the group field,
        ' here CITY. Could also write order by Sum (SubTotal)

SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

תוכנית A8:

```
Public Function A8()   'Basic Summary Query with Grouping and Requirements
                      ' on Output
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "GroupedOrders"
DoCmd.DeleteObject acQuery, strQueryName
```

```
strTableName = "Orders"
' Using AS to define an Alias, where [] are needed since a space is included
strSQL = "Select Sum(SubTotal) as SumSub, Avg(Shipping) as AveShip, " _
& " Count (Orders.OrderID) as [Num Orders], Min(IncomeTax) as MinTax, " _
& "Max(IncomeTax) as [Max Tax], City" & " From " & strTableName _
& " Group By City HAVING Count(City) > 1 Order By 1 desc" 'note HAVING is
            ' on a group function, while WHERE is on a record function

SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

אם ברצוננו לקבוע דרישות לגבי תוצאות הפונקציה, ולהציג רק את התוצאות שממלאות אחר הדרישות, נוסיף את הביטוי Having, שהוא דומה ל-Where, רק ש-Where עובד ברמת רשומה, ומסלק רשומות מסוימות מהחישובים, כאשר Having עובד על תוצאת פונקציה שהופעלה יחד עם הביטוי Group By, ומסלק שורות שכללו תוצאת חישוב שאינן ממלאות אחר הדרישה.



תרגיל 20:

בטבלת עובדים, מה ממוצע השכר וממוצע הציון במבחן הכניסה.

תרגיל 21:

בטבלת עובדים, חשב את ממוצע השכר וממוצע הציון במבחן הכניסה עבור עובדים מכל עיר. סדר את התוצאות לפי סדר שכר יורד.

תרגיל 22:

בטבלת עובדים, חשב את ממוצע השכר עבור עובדים מכל עיר. סדר את התוצאות לפי סדר שכר יורד. כלול רק ערים שממוצע הציון במבחן הכניסה הוא מעל 80.

תרגיל 23:

בטבלת עובדים, חשב את ממוצע השכר עבור עובדים מכל עיר שהשכר שלהם הוא מעל 5,000 ש"ח לחודש. סדר את התוצאות לפי עיר בסדר אלפביתי.

תרגיל 24:

כתוב שאילתה המבררת לכמה תלמידים יש ממוצע מעל 90, ולפחות הורה אחד שלמד פחות מ-10 שנים.

תוצאות ייחודיות וחלקיות

הוספת המילה DISTINCT לפקודת SELECT גורמת לתוצאה מסוימת להופיע פעם אחת בלבד. היא גם גורמת לכך שהתוצאות תהיינה מסודרות בסדר אלפביתי לפי השדה הימני ביותר. המילה DISTINCTROW אינה משפיעה על סדר התוצאות, ומונעת תוצאות כפולות בפלט אם כל השדות ברשומות הכפולות לכאורה, זהים. הדבר אפשרי אם מדובר על שאילתה בטבלאות קשורות, למשל לקוחות והזמנות, כשהפלט מתקבל מטבלת האב בלבד. למשל, כשתוצאת הפלט היא שם הלקוח עבור כל הזמנה. במקרה כזה שם הלקוח יופיע פעם אחת בלבד למרות שייתכן וביצע מספר הזמנות.

תוכנית A9: שימוש במילה DISTINCT.

```
Public Function A9()   'Distinct Keyword
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "DISTINCT"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Orders"
'  If you use DISTINCT, the cities will appear only once. But if you use
'  DISTINCTROW they will appear numerous times, because the row that they
'  are a part of is not distinct, even though the output is.
'  Also, you can't use distinct with order by, because distinct orders
'  If you display only city you will have even fewer results
strSQL = "Select Distinct Discount,City From " & strTableName

SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

שים לב שהתוצאות מסודרות לפי השדה הראשון. אם נרשום שני שדות: Discount ו-City תופענה 14 שורות בתוצאה, כשכל אחת שונה מחברתה, אך אם נרשום City בלבד, תופענה 10 רשומות בלבד (מתוך 19).

תוכנית A10: שימוש במילה DistinctRow.

```
Public Function A10()  'Alternative Join Query with DistinctRow
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
   strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "DistinctRow"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
' Table name prefix needed only if 2 tables have field with same name
' Distinctrow works here to prevent the display of identical records from Customers
strSQL = "Select DistinctRow CustomerLastName From "  & strTableName1 & ", " _
   & strTableName2 & " Where " & LinkPhrase & " Order BY CustomerLastName "
' This clashes with DISTINCT keyword
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

שם כל לקוח יופיע פעם אחת בלבד, משום שביקשנו ששדות מאותה רשומה יופיעו מספר פעמים וכאן נסתייע ב-DistinctRow. שים לב שמותר להשתמש בביטוי ORDER BY יחד עם DistinctRow שהרי DistinctRow אינו משפיע על הסדר.

לפעמים נרצה לראות רק מספר או אחוז מסוים מתוצאות השאילתה. במקרה כזה נרשום TOP n או TOP n PERCENT, בהתאמה. אם סדרנו את השאילתה לפי המפתח הרצוי, מילה זו תאפשר לנו, למשל, לראות את שלוש הרשומות הטובות ביותר (או הגרועות ביותר) בטבלה. לכן השימוש העיקרי של TOP הוא יחד עם ORDER BY.

תוכנית A11: דוגמה לשימוש במילה TOP.

```
Public Function A11()   'TOP Keyword
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SelectCust"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers1"
' alternative: strSQL = "Select TOP 3 IDCode, LastName etc.
strSQL = "Select top 50 Percent IDCode, LastName, Debt + _
   OldDebts as Balance From " & strTableName & " Order By Debt + OldDebts desc"
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
```



תרגיל 25:

על בסיס טבלת השאלות, נרצה להוציא רשימת מספרי זהות (מסודרת בסדר עולה) של התלמידים ששאלו ספרים, אך איננו רוצים שמספר זהות מסוים יופיע יותר מפעם אחת. בנה שאילתה מתאימה.

תרגיל 26:

על בסיס טבלת השאלות, נרצה להוציא רשימה אלפביתית של שמות ומספרי הזהות התלמידים ששאלו ספרים, אך איננו רוצים ששם תלמיד יופיע יותר מפעם אחת. בנה שאילתה מתאימה.

תרגיל 27:

בטבלת בית הספר, כתוב שאילתה המפיקה את שמות 5 הסטודנטים הבאים מהמשפחות הגדולות ביותר, ואת סכום שנות הלימוד של הורי כל אחד מחמשת הסטודנטים.

תרגיל 28:

בטבלת בית הספר, כתוב שאילתה המפיקה את שמות 5 הסטודנטים הבאים מהמשפחות הקטנות ביותר, ואת סכום שנות הלימוד של הורי כל אחד מחמשת הסטודנטים.

שאילתות בחירה ושאילתות פעולה

השאילתות שבהן דנו עד כה נקראות שאילתות בחירה, משום שהן מציגות חלק מהנתונים, או מפעילות על הנתונים פעולות אריתמטיות כלשהן, כולל סיכום, אך הנתונים המקוריים אינם משתנים.

שאילתות פעולה הן שאילתות המשפיעות על הנתונים והן מתחלקות לארבעה סוגים:

- **שאילתת עדכון** (Update query) משנה את תוכן כל הרשומות, או חלקן,
- **שאילתת יצירת טבלה** (Make Table query) הופכת את תוצאות השאילתה לטבלה חדשה,
- **שאילתת מחיקה** (Delete query) מוחקת חלק מהרשומות שבטבלה,
- **שאילתת הוספה** (append query) מוסיפה לטבלה רשומות מטבלה אחרת.

נגדיר כעת את פקודות SQL הנחוצות לשאילתות מסוגים אלה.

שאילתת עדכון - שאילתת עדכון מאפשרת לנו לשנות תוכן שדות כלשהם בטבלה אם תנאים מסוימים מתקיימים. המבנה הכללי של שאילתות מסוג זה הוא:

```
UPDATE  table SET  field =  expression [,field =  expression]... [WHERE condition]
```

שלוש הנקודות מציינות שהביטוי שלפניהן יכול לחזור כמה פעמים שנרצה. בנוסף, table יכולה להיות טבלה פשוטה, אך גם יכולה להיות טבלה שהיא תוצאה של קשר פנימי או חיצוני בין טבלאות.

תוכנית A12:

```
Public Function A12()  'Update Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                  ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "UpdateQuery"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Orders"
strSQL = "UPDATE Orders SET City = 'Cleveland', IncomeTax = 0 " _
              & " WHERE CustomerID = 24"
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

הביטוי UPDATE מודיע שמדובר בשאילתת עדכון, הביטוי SET מודיע שברצוננו לשנות את ערך השדות הרשומים בהמשך.

תוכנית A13: עדכון טבלאות קשורות.

```
Public Function A13()  'Update Joint Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
   strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "UpdateQuery"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
' if the key in both is based on 2 fields, use AND, e.g.
' if the key were first name and last name, the link phrase would be:
' Customers.FirstName = Orders.FirstName AND
' Customers.LastName = Orders.LastName
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
' Using Join to connect 2 tables
' Table name prefix needed only if 2 tables have field with same name
strSQL = "Update " & strTableName2 & " Inner Join " & strTableName1 _
    & " ON " & LinkPhrase & " Set Orders.City = 'ירושלים'" _
    & " Where Orders.City = 'Cleveland' and Customers.City =  'ירושלים'"
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

ניתן לעדכן שדות בכמה טבלאות. לאחר UPDATE במקום שם טבלה אחת נרשום:

```
table1 INNER JOIN table2 ON field1 = field2
```

ביטוי זה מחליף שם טבלה, שהרי גם קשר פנימי בין שתי טבלאות יוצר טבלה, ולכן הוא יכול להופיע בכל מקום ששם טבלה יכול להופיע.

שאילתת יצירת טבלה - שאילתה שיוצרת מהתוצאות טבלה, במקום להציגן בלבד. המבנה הכללי דומה לשאילתת בחירה, עם הוספת הביטוי INTO. המבנה הכללי הוא:

```
SELECT expressions INTO table FROM  tables [WHERE condition] [ORDER BY expression]
```

תוכנית A14:

```
Public Function A14()  'Double Join Make Table Query
'Making a new table based on information from 3 linked tables
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String, _
    strTableName3 As String, LinkPhrase1 As String, LinkPhrase2 As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "MakeTable"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
strTableName3 = "OrderItems"
LinkPhrase1 = "(" & strTableName1 & " INNER JOIN " & strTableName2 & _
           " ON Customers.CustomerID = Orders.CustomerID)"
LinkPhrase2 = "Orders.OrderID = OrderItems.OrderID"

' Table name prefix needed only if 2 tables have field with same name
strSQL = "Select Orders.OrderID, OrderDate, Orders.CustomerID, " _
        & "CustomerLastName, ItemNumber INTO NewTable From " _
        & LinkPhrase1 & " INNER JOIN " & strTableName3 & " ON " _
        & LinkPhrase2 & " Order BY Orders.OrderID "
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

בדוגמה זו בנינו טבלה בשם NEW TABLE מתוצאות הקשר שבין 3 טבלאות.

שאילתת מחיקה - שאילתה המוחקת רשומות מסוימות בהתאם לתנאים מסוימים המוגדרים בפקודת ה-SQL. המבנה הכללי של שאילתת מחיקה הוא:

```
DELETE  [table.* ]… FROM  table WHERE condition
```

תוכנית A15: דוגמה פשוטה של שאילתת מחיקה בטבלה אחת.

```
Public Function A15()   'Delete Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "DeleteCust"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "NewTable"
strSQL = "Delete From " & strTableName _
       & " Where OrderDate > #1/10/1995#" 'Uses American method for dates
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

די לכתוב את שם הטבלה ממנה נמחקות הרשומות, ואת התנאי למחיקה.

תוכנית A16: מחיקה מטבלאות קשורות.

```
Public Function A16()  'Delete using a Join Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "JoinDelete"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
' if the key in both is based on 2 fields, use AND, e.g.
' if the key were first name and last name, the link phrase would be:
' Customers.FirstName = Orders.FirstName AND
' Customers.LastName = Orders.LastName
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
' Table name prefix needed only if 2 tables have field with same name
strSQL = "Delete Orders.* From " & strTableName1 & " Inner Join " & strTableName2
& " ON " & LinkPhrase & " Where OrderDate > #1/10/97# and _
 Customers.City = 'ירושלים' "
```

```
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

לאחר DELETE נכתוב *.Orders כדי שנדע מאיזו טבלה מתוך שתי הטבלאות הקשורות נמחק רשומות.

שאילתת הוספה - שאילתה המאפשרת הוספת רשומות מטבלה אחת לאחרת. מבנה שתי הטבלאות אינו חייב להיות זהה, אך אם כן, תחביר הפקודה ב-SQL פשוט יותר.

כשמעתיקים מטבלה כלשהי לטבלה שנייה זהה במבנה, תחביר הפקודה הוא כדלהלן:

```
INSERT INTO table2 SELECT table1.* FROM table1 WHERE condition
```

אפשר להסתכל על פקודה זו כאילו היא כתובה כך:

```
INSERT INTO table2 subquery
subquery = SELECT table1.*  FROM table1 WHERE condition
```

במילים: אנו מוסיפים לטבלה השנייה את השורות שהוצאנו מהטבלה הראשונה בעזרת שאילתת המשנה.

תוכנית A17: הוספה פשוטה.

```
Public Function A17()   'Inserting records into existing table
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "InsertCust"
DoCmd.DeleteObject acQuery, strQueryName

strSQL = "Insert INTO JerCustomers Select Customers.* from Customers  " _
         & " Where City = 'ירושלים'"
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

במקרה הזנת שדות בטבלה כלשהי משדות או ביטויים אריתמטיים המבוססים על טבלה שנייה, התחביר המתאים הוא:

```
INSERT INTO table2 (field [,field]...) subquery
subquery = SELECT expression[,expression]...  FROM table1 WHERE condition
```

מספר השדות שלאחר השם table2 הוא כמספר הביטויים לאחר המילה SELECT. הביטויים יכולים להיות שילובי שדות ב-table1, או קבועים. פקודה זו גורמת להוספת רשומה לטבלה table2. מספר הרשומות המתווספות הוא כמספר הרשומות ב-table1 שמקיימות את התנאי המופיע לאחר המילה Where. כל ביטוי בשאילתת המשנה יחליף את ערך השדה המקביל ב-table2. כל שדה ב-table2 שאינו כלול בסוגריים, יקבל את ברירת המחדל שלו ברשומה שמתווספת. הדרך היחידה לשנות ערך שדה הוא לכלול אותו בסוגריים ולרשום את הביטוי המחליף בשאילתת המשנה.

תוכנית A18: דוגמה לשאילתת הוספה מורכבת.

```
Public Function A18()  'Insert using Join Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String, LinkPhrase As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "JoinInsert"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "Customers"
strTableName2 = "Orders"
LinkPhrase = "Customers.CustomerID = Orders.CustomerID"
' Using Join to connect 2 tables
' Table name prefix needed only if 2 tables have field with same name
' Any field which is not included will get blank value,
' can put in constants or expressions
strSQL = "Insert into NewTable (OrderID,OrderDate,CustomerID,ItemNumber) " _
    & "Select OrderID + 1, OrderDate +17, Orders.CustomerID, 6 From " _
    & strTableName1 & " Inner Join " & strTableName2 & " ON " & LinkPhrase _
    & " Where OrderID > 25 "
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```



תרגיל 29:

ההנהלה החליטה לתת העלאת משכורת של 10% לכל עובד שקיבל מעל 80 במבחן הכניסה. כתוב שאילתת עדכון למטרה זו.

תרגיל 30:

הוחלט שתלמידים הבאים ממשפחות גדולות (מעל 3 ילדים), זכאים להארכה של שבוע מעל משך הזמן הנקוב בטבלת ההשאלות. עדכן את טבלת ההשאלות בצורה מתאימה.

תרגיל 31:

בשל ריבוי טעויות בטבלת השאלות, הוחלט לכלול גם את השם הפרטי ושם המשפחה של כל שואל. בנה טבלה חדשה בעזרת שאילתת יצירת טבלה שבונה טבלה שתיקרא טבלת השאלות מתוקנת. אחר, מחק את טבלת ההשאלות הישנה.

תרגיל 32:

בטבלת ההשאלות מתוקנת נרצה למחוק את כל ההשאלות שהסתיימו. נעביר תחילה את ההשאלות שהסתיימו לטבלה חדשה בשם **טבלת היסטוריית השאלות**, ואחר נמחק אותן מטבלת ההשאלות המתוקנת.

תרגיל 33:

נניח שהוחזרו ספרים נוספים ועודכנו כראוי ב**טבלת השאלות מתוקנת**. כתוב שאילתה המעבירה את הרשומות המתאימות מ**טבלת השאלות מתוקנת** ל**טבלת היסטוריית השאלות**, ואחר מחק את הרשומות מ**טבלת השאלות מתוקנת**.

שאילתות איחוד

שאילתת איחוד (Union Query) - שאילתה המורכבת ממספר שאילתות SELECT. התוצאה היא טבלה אחת עם נתונים מכל הטבלאות המיוחסות. מספר השדות הנקלטים בכל שאילתה חייב להיות זהה, ומבנה השדות המקבילים בכל שאילתה חייב להיות זהה. השדות המקבילים יכולים להיות ביטויים אריתמטיים, אך סוג התוצאה חייב להיות זהה לסוג השדות המקבילים.

כדאי לציין שהטבלאות המקוריות אינן חייבות להיות זהות, רק תוצאות השאילתות חייבות להיות זהות מבחינת מספרי וסוגי השדות. השמות בפלט יהיו השמות שבהם משתמשים בשאילתה הראשונה, אך ניתן להשתמש גם ב**כינוי** (Alias) (בעזרת מילת העזר AS). ניתן להוסיף את הביטוי ORDER BY field כאשר field הוא שם שדה המשתתף בשאילתה הראשונה.

המבנה הכללי של הפקודה הוא:

```
SELECT expression [,expression]... FROM  table1 [WHERE condition]
UNION
SELECT expression [,expression]... FROM  table2 [WHERE condition]
UNION [ALL]
SELECT expression [,expression]... FROM  table3 [WHERE condition]
.
.
[ORDER BY expression]
```

תוכנית A19: דוגמה לשאילתת איחוד.

```
Public Function A19()   'Union Query
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName1 As String, strTableName2 As String
On Error Resume Next 'needed if the query we wish to delete doesn't
                     ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "UnionCust"
DoCmd.DeleteObject acQuery, strQueryName
strTableName1 = "IntCustomers"
strTableName2 = "Customers"
' writing UNION ALL instead of UNION will also give duplicates
' ORDER BY must be last, use names of first, which appear in output
strSQL = "Select CustomerID, FirstName, SurName, CityCountry From " _
         & strTableName1 _
         & " Union " _
         & "Select CustomerID, CustomerFirstName, CustomerLastName, " _
         & "Rtrim (City) & ', Israel' From " & strTableName2 _
         & " Order By CustomerID"
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

הוספת המילה ALL גורמת לכך שגם שורות כפולות תופענה.

סיכום פקודות SQL (שני הביטויים האחרונים יוסברו בהמשך):

סוג שאילתה	באנגלית	פקודת SQL
בחירה	Select	SELECT
יצירת טבלה	Make Table	SELECT...INTO
עדכון	Update	UPDATE...SET
מחיקה	Delete	DELETE...FROM
הוספה	Append	INSERT INTO...SELECT
הגדרת טבלה	Define Table	CREATE TABLE
מחיקת טבלה	Deleting Tables	DROP TABLE
בניית אינדקס	Define Index	CREATE INDEX
מחיקת אינדקס	Deleting Index	DELETE INDEX



תרגיל 34:
חברה שמקבלת לרשותה את טבלת **בית ספר** וטבלת **עובדים** מעוניינת להכין רשימה אלפביתית של כל השמות והכתובות שמופיעים בטבלאות. הכן שאילתת איחוד מתאימה.

שאילתות משנה

שאילתת משנה (Subquery) - שאילתה שמופיעה כחלק משאילתה אחרת. שאילתת משנה שתוצאתה היא ערך קבוע, יכולה להחליף ערך קבוע בשאילתה המקורית, ושאילתת משנה שתוצאתה טבלה, יכולה להחליף טבלה בשאילתה המקורית. שאילתת משנה חייבת להופיע בסוגריים.

תוכנית A20: דוגמה להחלפת קבוע בתת-שאילתה.

```
 Public Function A20()   'SubQueries
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String, strSubQuery As String

' On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SubQuery"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers1"
```

```
' SubQuery substitutes for a value
strSubQuery = "(Select avg(Debt + OldDebts) From " & strTableName & ")"
strSQL = "Select IDCode, LastName, Debt + OldDebts as Balance From " _
        & strTableName _
        & " Where (Debt + OldDebts) > " & strSubQuery
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

שאילתת המשנה היא שאילתת סיכום שמספקת כתוצאה ערך קבוע אחד, ממוצע סכום השדות: Debt + OldDebts.

תוכנית A21: דוגמה להחלפת רשימת ערכים בשאילתת משנה.

```
Public Function A21()   'SubQueries2
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
   strTableName As String, strSubQuery As String

' On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SubQuery2"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers"
' This query finds all customers who have made more than one order.
' If you use NOT IN instead of IN you have all customers who didn't make more
' than 1 order
' SubQuery produces a subset from which main query chooses

strSubQuery = "(Select CustomerID From Orders  Group By CustomerID " _
             & "HAVING Count(CustomerID)>1)"
strSQL = "Select * From " & strTableName _
          & " Where CustomerID In " & strSubQuery
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

האופרטורים שפועלים על רשימת ערכים הם: האופרטור IN שמחזיר ערך True אם ערך שדה מסוים מופיע ברשימה, ו-NOT IN, שמחזיר ערך True אם ערך שדה מסוים אינו מופיע ברשימה.

תוכנית A22:

```
Public Function A22()   'SubQueries3
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String, strSubQuery As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                  ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SubQuery3"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers"
' Another way to do same as last query using EXISTS
' This query finds all customers who have made more than one order,
'  if you use EXISTS, and all customers who didn't make more
' than 1 order if you use NOT EXISTS
' SubQuery produces a subset from which main query chooses

strSubQuery = "(Select CustomerID From Orders  " _
          & "Where orders.customerid = customers.customerid " _
          & "Group By CustomerID HAVING Count(CustomerID)>1)"
strSQL = "Select * From " & strTableName _
         & " Where Not Exists " & strSubQuery
SelectTable strSQL, strQueryName, booGood
If booGood Then
  DoCmd.OpenQuery strQueryName
Else
  MsgBox "Problem"
End If
End Function
```

האופרטור EXISTS שווה TRUE אם קיימת לפחות רשומה אחת בתוצאת שאילתה כלשהי. לכן, שאילתה זו תחזיר את כל הלקוחות שביצעו 0 או 1 הזמנות, כלומר כל הלקוחות שעבורם שאילתת המשנה אינה מחזירה רשומה, משום שלאותו לקוח אין הקבצת הזמנות המורכבת משתי רשומות ויותר.

תוכנית A23:

```
Public Function A23()   'SubQueries4
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String, strSubQuery As String

On Error Resume Next 'needed if the query we wish to delete doesn't
                     ' exist so that no error will occur.
' erase previous query having this name, if they exist
strQueryName = "SubQuery4"
DoCmd.DeleteObject acQuery, strQueryName
strTableName = "Customers1"
' This query finds all customers whose
' debt > (Olddebts for at least one person) (ANY or SOME)
' If you use ALL you find all customers whose
' debt > (Olddebts for all people)
' SubQuery produces a subset from which main query sees
' if debt in each row is more than olddebts in ALL rows or ANY rows,
' depending on what you wrote.
strSubQuery = "(Select OldDebts From Customers1)"
strSQL = "Select * From " & strTableName & " Where Debt > ANY " & strSubQuery
SelectTable strSQL, strQueryName, booGood
If booGood Then
   DoCmd.OpenQuery strQueryName
Else
   MsgBox "Problem"
End If
End Function
```

גם במקרה זה אנו מוציאים כפלט רשימת ערכים (פה עבור OldDebts), ומשווים את הרשימה לשדה Debt בעזרת ANY (או SOME או ALL). ANY יחזיר ערך TRUE לתנאי, גם אם רק ערך אחד ברשימה קטן מ-Debt. ALL יחזיר ערך TRUE לתנאי, רק אם כל הערכים ברשימה קטנים מ-Debt. בכל פעם שהתנאי מתמלא (מתקבל ערך TRUE), הרשומה תיכלל בתוצאות הפלט.



תרגיל 35:

ברצוננו לבדוק מיהם הסטודנטים (שמות) ששואלים ספרים רבים יותר מהסטודנט הממוצע. כתוב שאילתה מתאימה.

תרגיל 36:.

ברצוננו לדעת שם, כתובת ומספר שנות לימוד האב והאם של סטודנט הרשום ב**טבלת השאלות מתוקנת** כאחד ששאל מעל ספר אחד.



תרגיל 37:

מסתבר שציון מבחן הכניסה לעובד מתואם בערך לציון הממוצע שנתי של סטודנט. כתוב שאילתה שמאתרת את העובדים שציונם במבחן כניסה מעל לציון הממוצע של כל תלמיד, וכתוב שאילתה שנייה המאתרת את העובדים שהציון שלהם במבחן כניסה הוא מעל לציון הממוצע של לפחות תלמיד אחד.

מחיקת רשומות מטבלה

אפשר למחוק רשומות בעזרת שיטה (Method) מיוחדת לקבוצות רשומות.

תוכנית A24: דוגמה לשיטת המחיקה, Delete.

```
Function A24()
' Deleting records can be done for dynasets or table recordsets
Dim MyDb As Database, MyTable As Recordset
Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MyTable = MyDb.OpenRecordset("Customers", dbOpenTable)
With MyTable
.MoveFirst
Do Until .EOF
  If ![RegionalCode] = 56789 Then
    MsgBox !CustomerLastName & "  " & !RegionalCode
    .Delete
  End If
  .MoveNext
Loop
.Close
End With

End Function
```



תרגיל 38:

הוחלט לפטר עובדים (כלומר למחוק אותם מהטבלה), אם מתקיימים שני תנאים:

- ציונם במבחן כניסה למטה מ-80
- משכורתם מעל 5,000 שקל.

כתוב שיגרה שתמלא דרישה זו.



תרגיל 39:

בית הספר מקבל מנהל חדש שרוצה להעלות את הרמה. הוא מחליט להפסיק לימודי (למחוק מהטבלה) תלמיד אם מתקיימים עבורו שני תנאים:

- ממוצע למטה מ-70.
- סכום של מעל 20 שנות לימודים לשני הוריו.

כתוב שיגרה המבצעת את הפעולה הדרושה.

מיון בטבלה

כזכור, בקבוצת רשומות שסוגה מוגדר כ dbOpenTable: אפשר להשתמש במפתחות ואינדקסים. אינדקס הוא טבלה שמחזיקה את סדר הרשומות לפי שדה כלשהו - שדה שעבורו בונים אינדקס. למשל, בטבלת Customers המפתח הראשי הוא CustomerID, אך ייתכן ונרצה אינדקס לפי שם משפחה. האינדקס יוכל להיראות כך:

שם משפחה	מספר לקוח
בראון	110
בראון	111
גרינברג	23
גרינברג	37
גרינברג	35
הלוי	26
חיימוביץ	24
חיימסון	55
חנני	31
כהן	28
לבני	30
נחמן	76
נחמני	22
פרימן	27
רון	25
רון	36
רון	32

בצורה דומה ניתן לבנות אינדקסים עבור כל שדה. אינדקסים אלה יכולים להיות ייחודיים (Unique), כלומר, כל ערך של האינדקס חייב להיות שונה מחברו, או שהם יכולים להיות לא ייחודיים, דהיינו, אפשור ערכים כפולים. בדוגמה הקודמת למשל, היו מספר שמות זהים. אינדקס גם יכול להיות מורכב ממספר שדות, למשל, אינדקס המורכב משם משפחה ושם לקוח יכול להיראות כך:

שם משפחה	שם פרטי	מספר לקוח
בראון		111
בראון	חיים	110
גרינברג	חיים	23
גרינברג	חיים יעקב	37
גרינברג	שלום	35
הלוי	לאה	26
חיימוביץ	צבי	24
חיימסון		55
חנני	שלום	31
כהן	אביאל	28
לבני	חננאל	30
נחמן	גברי	76
נחמני	שמואל	22
פרימן	ראובן	27
רון	רות	25
רון	שלמה	32
רון	שלמה צבי	36

בשפת SQL הפקודה לבניית אינדקסים נראית כך:

```
CREATE [UNIQUE] index name ON table name (field name list)
```

המילה UNIQUE מגדירה את האינדקס כייחודי. שם האינדקס יהיה index name עבור הטבלה table name. השדות הכלולים באינדקס יופיעו בסוגריים.

תוכנית A25: מחיקת אינדקס קודם בשם זה (אם היה) ובניית אינדקס חדש.

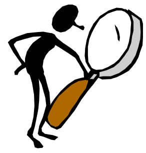
```
Public Function A25()  'Create Index
Dim strSQL As String, booGood As Boolean, strQueryName As String, _
    strTableName As String, strIndexName As String
Dim db As Database
Set db = CurrentDb()

On Error Resume Next 'needed if the query we wish to delete doesn't
                    ' exist so that no error will occur.
' erase previous query and table having this name, if they exist
strQueryName = "CreateIndex"
strTableName = "Customers"

DoCmd.DeleteObject acQuery, strQueryName
strIndexName = "Name"
' erase an index using SQL
strSQL = "Drop Index " & strIndexName & " ON " & strTableName
db.Execute strSQL
' SelectTable strSQL, strQueryName, booGood
' If booGood Then
'   DoCmd.OpenQuery strQueryName
' Else
'    MsgBox "Problem"
' End If

' Define new index name and query name
strSQL = "CREATE INDEX " & strIndexName & " ON " & strTableName & _
    " (CustomerLastName, CustomerFirstName)"

db.Execute strSQL
' SelectTable strSQL, strQueryName, booGood
' If booGood Then
 '     DoCmd.OpenQuery strQueryName
 '  Else
 '     MsgBox "Problem"
 '  End If
End Function
```



הערות:

- אם אין מפתח ראשי (או שמחקת אותו בעזרת DROP INDEX), אפשר לבקש שהאינדקס שנוסף יהיה מפתח ראשי, על ידי הוספת המילים WITH PRIMARY בסוף הפקודה.
- אם מוסיפים WITH DISALLOW NULL וערך האינדקס הוא NULL, אי אפשר להוסיף רשומה. אם מוסיפים WITH IGNORE NULL, אפשר להוסיף רשומה גם כאשר ערך האינדקס הוא NULL, אך זיהוי רשומה זו לא יהיה באינדקס.

תוכנית A26: שימוש במפתח הראשי ומפתח משני (האינדקס).

```
Function A26()

' Printing the file in the order of the primary key
' and the secondary key
Dim MyDb As Database, MyTable As Recordset
' Set mydb = DBEngine.Workspaces(0).Databases(0)
Set MyDb = CurrentDb()
Dim TableName As String
TableName = "Customers"
Set MyTable = MyDb.OpenRecordset(TableName, dbOpenTable)

Debug.Print "Without Index, just primary key" ' default is primary index

A: With MyTable
Do Until .EOF
    Debug.Print !CustomerID; Tab(10); !CustomerLastName _
    ; Tab(25); !CustomerFirstName
    .MoveNext
Loop
B:
.Index = "Name"
Debug.Print "With Index"
.MoveFirst
Do Until .EOF
    Debug.Print !CustomerID; Tab(10); !CustomerLastName _
    ; Tab(25); !CustomerFirstName
    .MoveNext
Loop
.Close
AA: End With
End Function
```

הסבר: שורה A ושורה AA: הפקודה WITH tablevariable.....END WITH מאפשרת לדלג על משתנה הטבלה בתוך טווח הפקודה, בכל פעם שמוזכר שם שדה בטבלה או שיטה (Method) המופעלת על הטבלה. אפשרות זו משפרת את קריאות השיגרה.

שורה B: הפעלנו את האינדקס NAME, שמסדר את הטבלה לפי שם משפחה ושם פרטי. הלולאה בפקודה הבאה מדפיסה את נתוני הטבלה לפי סדר האינדקס NAME. אך ראשית יש צורך למקם את המצביע ברשומה הראשונה שמופיעה באינדקס.

כמו שהוזכר, אפשר להשתמש באינדקסים רק בקבוצות רשומות מסוג טבלה. בקבוצות רשומות מסוג קבוצה דינמית או תצלום בזק, אפשר להשתמש במאפיין SORT של קבוצת הרשומות.

תוכנית A27: מיון נתוני טבלה ב-3 סדרים שונים.

```
Function A27()

' Sorting. Open as a dynaset which adopts order of the
' table it's based on. Then change sort order and open second dynaset
' based on first dynaset.

Dim MyDb As Database, MyTable As Recordset, MyTable2 As Recordset
' Set mydb = DBEngine.Workspaces(0).Databases(0)
Set MyDb = CurrentDb()
Dim TableName As String
TableName = "Customers"
Set MyTable = MyDb.OpenRecordset(TableName, dbOpenDynaset)

A: Debug.Print "Without Sort Property" ' default is primary index

With MyTable
Do Until .EOF
      Debug.Print !CustomerID, !CustomerLastName, !CustomerFirstName
      .MoveNext
Loop
B:
.Sort = "[City]"
End With
B1: Set MyTable2 = MyTable.OpenRecordset
Debug.Print "With Sorting on City"
With MyTable2
C: Do Until .EOF
      Debug.Print !CustomerID, !CustomerLastName, _
      !CustomerFirstName, !City, !RegionalCode
      .MoveNext
```

```
Loop
.Close 'we must close Mytable2, because we are going to use it again.
End With
D: MyTable.Sort = "[RegionalCode] desc, [CustomerLastName]"
Set MyTable2 = MyTable.OpenRecordset
With MyTable2

Debug.Print "With Sorting on RegionalCode and LastName"
Do Until .EOF
      Debug.Print !CustomerID, !CustomerLastName, _
       !CustomerFirstName, !City, !RegionalCode
      .MoveNext
Loop
.Close
End With
MyTable.Close

End Function
```

שורה A: כאשר מגדירים קבוצה דינמית על בסיס טבלה, הסדר המקורי של הקבוצה הדינמית הוא לפי סדר המפתח הראשי של הטבלה שעליה מבוססת הקבוצה.

שורה B: קביעת המאפיין SORT של קבוצה דינמית תגרום לכך שכאשר יוצרים קבוצת רשומות חדשה מאותה קבוצה דינמית, המיון יתבצע לפי הערך שניתן למאפיין SORT.

שורה B1: פקודת OpenRecordSet ללא ארגומנטים. היות והאובייקט הוא קבוצה דינמית, סוג קבוצת הרשומות שנוצרה הוא גם קבוצה דינמית.

שורה C: מעבר דרך הקבוצה הדינמית לפי הסדר החדש. בסוף התהליך נסגור את הקבוצה הדינמית, כדי שנוכל להשתמש בה שוב.

שורה D: קביעת המאפיין SORT של הקבוצה הדינמית לשני מפתחות מיון, הראשון בסדר יורד, ושוב יצירת קבוצה דינמית חדשה.

תוכנית A28: דוגמה לעבודה עם תצלום בזק.

```
Function A28() ' SnapShots
Dim MyDb As Database
Dim MySnapshot As Recordset, Snap2 As Recordset, Snap3 As Recordset
Dim subsc As Integer
Dim S1 As Variant, s2 As String, s3 As String
Set MyDb = DBEngine.Workspaces(0).Databases(0)
' snapshots and dynasets can be opened from queries as well as
' tables. Otherwise they are like regular datasets. A snapshot
```

```
' cannot update the query that it is based on, nor the tables that the
' the query is based on, but a dynaset can. However, a snapshot can
' be sorted and placed into a different snapshot. Using snapshots is
' more efficient than using dynasets or tables, because you are
' not allowing updating
S1 = "Select CustomerID,CustomerLastName,MoneyOwed,DateLastPAyment,City From _
      Customers"
A: Set MySnapshot = MyDb.OpenRecordset(S1, dbOpenSnapshot)
' note: s1 must be variant, because null is a variant type, and S1 may hold it
With MySnapshot
MsgBox "Set 1: Order is by CustomerID, the Primary Key order"
For subsc = 1 To 3
B: S1 = IIf(![DateLastPayment] = Null, " ", ![DateLastPayment])
   s2 = Str$(![MoneyOwed])
   s3 = CStr(![CustomerLastName])

   MsgBox S1 & s2 & "  " & s3 & "  " & !CustomerID
   .MoveNext

Next subsc
C:
.Sort = "[CustomerLastName]"
' sorting and filtering can only be done on dynasets
' and snapshots

Set Snap2 = MySnapshot.OpenRecordset(dbOpenSnapshot)
.Close
End With
With Snap2
MsgBox " Set 2"
For subsc = 1 To 3
   S1 = IIf(![DateLastPayment] = Null, " ", ![DateLastPayment])
   s2 = Str$(![MoneyOwed])
   s3 = CStr(![CustomerLastName])
   MsgBox S1 & s2 & "  " & s3
   .MoveNext

Next subsc

D:
' sorting and filtering the new set
.Sort = "[CustomerLastName] Desc"
.Filter = "[City]='תל אביב'"  'set filter condition
```

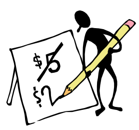
```
Set Snap3 = Snap2.OpenRecordset(dbOpenSnapshot)
.Close
End With
With Snap3
MsgBox " Set3"
For subsc = 1 To 3
  S1 = IIf(![DateLastPayment] = Null, " ", ![DateLastPayment])
  s2 = Str$(![MoneyOwed])
  s3 = CStr(![CustomerLastName])
  MsgBox S1 & s2 & "  " & s3 & "  " & !City
  .MoveNext
Next subsc
.Close
End With
End Function
```

שורה A : דרך נוספת להגדיר קבוצת רשומות היא על ידי הכללת שורת SQL בשורת הגדרת הקבוצה. בלולאה הראשונה סדר ההצגה הוא לפי סדר המפתח הראשי בטבלה שעליה מבוססת שאילתת הגדרת תצלום הבזק.

B : היות והשדה DateLastPayment יכול להיות ריק (Null), אנו מגדירים את המשתנה S1 כסוג Variant, כדי שיוכל להחזיק ערך Null, אם נחוץ.

C : הגדרת סדר מיון חדש, שישתקף בקבוצת רשומות שתיווצר מתצלום בזק זה, כפי שאפשר לראות מלולאת הפלט הבא.

D : הגדרת סדר מיון יורד, והגדרת מסנן שיציג רשומות שמקיימות את דרישת הסינון.



תרגיל 40:

כפי שכבר הוזכר, בטבלת הספריה המפתח הוא הכותר והמחבר, יחד. בנוסף, אנו מעוניינים במפתחות המשנה הבאים:

- מספר סידורי (ייחודי),
- ISBN (ייחודי),
- שם המחבר בלבד (לא ייחודי),
- מוציא לאור בלבד.

הוסף אינדקסים אלה, והדפס רשימת ספרים מסודרת לפי כל אחד מהאינדקסים.



תרגיל 41:

בנה שאילתה על בסיס הטבלאות **טבלת השאלות מתוקנת**, טבלת **סטודנטים** וטבלת **ספריה** שמציגה את שם הספר, שם הסטודנט, ומספר הילדים במשפחה, וממיינת אותם לפי מספר הילדים בסדר יורד, ולפי שם ספר בסדר עולה. אפשר למשתמש למחוק את כל הרשומות של משפחות עם מספר ילדים מסוים שהוא עצמו יקבע.

תרגיל 42:

בנה צילום בזק שייאפשר להציג את הנתונים שהוצגו בתרגיל הקודם.

חיפוש בטבלה

כפי שכבר ראינו, מיון קבוצת רשומות משתנה לפי הסוג - מיון סוג טבלה מתבצע בעזרת אינדקס, והסוגים האחרים נעזרים במאפיין SORT. גם חיפוש משתנה לפי הסוג. נתחיל בקבוצות רשומות דינמיות ותצלומי בזק. חיפוש מתבצע בעזרת הפקודה FindNext. המבנה הכללי של Findnext הוא: FINDNEXT condition, כאשר condition הוא מחרוזת המחזיקה את תנאי החיפוש. כלומר אנו מחפשים רשומה שמקיימת את הדרישה המוגדרת במחרוזת.

תוכנית A29: חיפוש שדות שלא מולאו, בעזרת FindNext.

```
Function A29()
' determines which records do not have the fax number filled in
Dim MyDb As Database, MyTable As Recordset
Set MyDb = DBEngine.Workspaces(0).Databases(0) ' or currentdb ()
Set MyTable = MyDb.OpenRecordset("Customers", dbOpenDynaset)
' the findnext function works only with dynasets and snapshots
' we could have defined criterion = "[FaxNumber] = Null"
' and then written: MyTable.findnext criterion.
' With table recordsets use the seek method, which can specify an
' equality or inequality criterion on the presently activated
' index only.
With MyTable
A:
.FindNext "[FaxNumber] = Null"
Do Until .NoMatch
   MsgBox ![CustomerID] & "  " & !CustomerLastName
   .FindNext "[FaxNumber] = Null"
Loop
.Close
End With
End Function
```

שים לב שכל שימוש ב-FindNext מאתר רק את המופע הבא, ולכן חייבים להפעיל FindNext מחדש לאחר כל איתור מוצלח. כאשר החיפוש מגיע לסוף הטבלה, המאפיין MyTable.Nomatch הופך להיות True, והלולאה נגמרת. בשורה A מוצגת הפקודה FindNext יחד עם הגדרת הדרישה הקובעת את החיפוש.

תוכנית A30:

```
Function A30() ' FindNext together with editing
' this finds all nulls and changes to what you want
Dim MyDb As Database
Dim Custds As Recordset
Dim msg As String
Set MyDb = DBEngine.Workspaces(0).Databases(0)
' only a dynaset or table can be updated. But findnext can
' be used with dynasets or snapshots.
Set Custds = MyDb.OpenRecordset("Customers", dbOpenDynaset)
With Custds
.FindNext "[FaxNumber] = Null"  'we could also automatically put in a number
                        ' for each null
Do Until .NoMatch
  .Edit
  msg = "Enter a fax number for " & ![CustomerLastName]
  ' the input function inputs the value you write in the box
  ' to the variable on the left
  ![FaxNumber] = InputBox(msg)
  .UPDATE
  .FindNext "[FaxNumber] = Null"
Loop
.Close
End With
End Function
```

תוכנית זו היא הרחבה של התוכנית הקודמת, אך במקום לחפש מספרי פקס שלא הוזנו, בתוכנית זו נוכל גם להזינם מיידית בעזרת עריכת הרשומות הקיימות בטבלה.

FindNext (וגם FindFirst, FindPrevious, FindLast)

השיטה FindNext (כמו גם השיטות FindPrevious, FindFirst, FindLast) פועלת בתיאום עם המאפיין Sort, שהרי אם הכוונה היא לאתר את המופע הבא, השאלה היא תמיד לפי איזה סדר, והתשובה היא לפי המיון שהוגדר לקבוצה. אם הקבוצה מבוססת על טבלה, המיון המקורי הוא לפי המפתח הראשי שלה. אם היא מבוססת על שאילתה, הסדר המקורי הוא לפי סדר הרשומות בתוצאת השאילתה. אם השאילתה כוללת את הביטוי Order By, הסדר הוא לפי העמודות שהוזכרו בביטוי, אחרת, הסדר הוא לפי סדר עולה בעמודות, החל מהעמודה הימנית ביותר וכלה בשמאלית.

כאמור, כל זה אמור לגבי קבוצות דינמיות ותצלומי בזק. בקבוצות מסוג טבלה, הסדר נקבע לפי המפתח והאינדקס וכן גם החיפוש. אם נבצע חיפוש בקבוצה מסוג טבלה על שדה שאינו אינדקס, נבסס עליה קבוצה דינמית או תצלום בזק ונשתמש ב-FindNext. אך אם החיפוש הוא בשדה מפתח או אינדקס, אפשר להשתמש בשיטה הנקראת Seek. שיטה זו מורכבת משני שלבים:

1. בוחרים את האינדקס שיופעל בעזרת פקודה זו:

```
recordset.INDEX = "index name"
```

2. רושמים פקודת SEEK, שנראית כדלהלן:

```
SEEK operator, value,
```

כאשר operator הוא שוויון או אי שוויון ("=", ">", "<", ">=", "<=") ו-value הוא המשתנה או הקבוע הנחוץ להגדרת הדרישה. אנו מחפשים, איפוא, את הרשומה הראשונה (לפי סדר האינדקס שבחרת בשלב 1) שמקיימת את השוויון או האי-שוויון דלהלן:

```
index operator value.
```

index הוא משלב 1, operator ו-value הם משלב 2.

תוכנית A31:

```
Sub A31_CheckCustomer(ByVal Mispar_Lakuach, ByVal Cost)
' Seek can only be used for tablesets, and can only be an
' equality or inequality on the index field
Dim MyDb As Database, MyTable As Recordset
Set MyDb = CurrentDb()
Set MyTable = MyDb.OpenRecordset("Customers", dbOpenTable)
With MyTable
A:
.Index = "PrimaryKey"
B:
.Seek "=", Mispar_Lakuach
If .NoMatch Then
   Debug.Print Mispar_Lakuach  ' we could have sent a variable
                               ' back to the first program
   Debug.Print "There is an error"
Else
   .Edit
   If ![MoneyOwed] + Cost > ![CreditLimit] Then
        MsgBox "Can't fill this order for customer" & Str(Mispar_Lakuach)
   Else
     ![MoneyOwed] = ![MoneyOwed] + Cost
      Debug.Print ![MoneyOwed], Cost, Mispar_Lakuach
```

```
  End If
  .UPDATE
  End If
.Close       ' closing the table
End With
End Sub
```

בדוגמה זו אנו מעיינים בהזמנות לפי הסדר ומוסיפים את התת-סכום שבהזמנה לחוב הכולל של הלקוח. כדי לעשות זה, אנו קוראים לפונקציה שמפעילה פקודת Seek.

שורה A : שלב 1.

שורה B : שלב 2.

תוכנית A32: דוגמה לעדכון אקראי של טבלת לקוחות על בסיס טבלת תנועות.

```
Function A32() ' Adding Transactions
Dim MyDb As Database
Dim MyLak As Recordset, MyTrans As Recordset
Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MyTrans = MyDb.OpenRecordset("Transactions", dbOpenTable)
Set MyLak = MyDb.OpenRecordset("Customers", dbOpenTable)
With MyLak
.Index = "PrimaryKey"
Do Until MyTrans.EOF
  .Seek "=", MyTrans![CustomerID]
  ' use mylak.nomatch to test to see if the match didn't work
  ' if it didn't, you can check if he wants to add, else check
  ' for change or delete
  If MyTrans![TransactionType] = "a" And .NoMatch Then
    .AddNew          'adds blank new record
    ![CustomerID] = MyTrans![CustomerID]
    ![CustomerLastName] = MyTrans![CustomerLastName]
    .UPDATE  ' updates the change made to the new record
  ElseIf MyTrans![TransactionType] = "d" And Not .NoMatch Then
      MyLak.Delete
  Else: MsgBox "Error " & MyTrans!CustomerID
  End If
MyTrans.MoveNext    ' going to the next record
Loop
MyTrans.Close        ' closing the table
.Close
End With
End Function
```

הסבר: בתוכנית קיימים שני סוגי תנועות: אחד המצביע על הוספת לקוחות חדשים (קוד "a"), ושני המצביע על מחיקת לקוחות ("d"). נעבור דרך התנועות אחת-אחת בעזרת הפקודה MyTrans.MoveNext עד לסוף טבלת התנועות (MyTrans.EOF). נפעיל Seek על טבלת לקוחות לפי המפתח הראשי, ונשווה אותו למספר הלקוח שנמצא בתנועה. אם פקודת Seek אינה מצליחה למצוא רשומה בעלת המפתח הדרוש (NoMatch = True), וקוד הפעולה הוא a עבור הוספה, ניתן להוסיפה על בסיס נתונים בתנועה. אם פקודת Seek מאתרת רשומה בעלת מפתח דרוש (NoMatch = False) וקוד הפעולה הוא d עבור מחיקה, ניתן למחוק אותה מטבלת לקוחות.

תוכנית A33: דוגמה מסכמת לעדכון סדרתי.

```
Function A33() ' transactions when dealing with sequential files
' this gives the general idea, but doesn't handle either file after
' it hits the eof, nor does it handle error conditions.
Dim MyDb As Database
Dim MyLak As Recordset, MyTrans As Recordset, MyNew As Recordset, _
MyTrans2 As Recordset
Dim Custnum, Transnum As Integer
Dim NeedTrans, NeedCust As Integer
Set MyDb = DBEngine.Workspaces(0).Databases(0)
Set MyTrans2 = MyDb.OpenRecordset("Transactions", dbOpenDynaset)
Set MyLak = MyDb.OpenRecordset("Customers", dbOpenTable)
A: NewCustomers "NCustomers"  'Creates a new file called NCustomers
Set MyNew = MyDb.OpenRecordset("NCustomers", dbOpenTable)
B: MyLak.Index = "PrimaryKey"
C: MyTrans2.Sort = "[CustomerID]"
Set MyTrans = MyTrans2.OpenRecordset(dbOpenDynaset)
D: MyLak.MoveFirst
E: MyTrans.MoveFirst

Transnum = MyTrans![CustomerID]
Custnum = MyLak![CustomerID]
With MyNew
Do Until Transnum = 9999 And Custnum = 9999

   If Custnum < Transnum Then
      .AddNew
      ![CustomerID] = MyLak![CustomerID]
      ![CustomerLastName] = MyLak![CustomerLastName]
      ![MoneyOwed] = MyLak![MoneyOwed]
      ![CreditLimit] = MyLak![CreditLimit]
      .UPDATE
      NeedCust = 1
   ElseIf Custnum > Transnum Then
     NeedTrans = 1
     If MyTrans![TransactionType] = "a" Then
```

```
      .AddNew
      ![CustomerID] = MyTrans![CustomerID]
      !CustomerLastName = MyTrans!CustomerLastName
       .UPDATE
      Else
        Debug.Print "Add Error"
      End If
   Else 'means they are equal
      If MyTrans![TransactionType] = "c" Then 'c means change
      ' d will automatically be deleted
         .AddNew
         ![CustomerID] = MyTrans![CustomerID]
         ![CustomerLastName] = MyLak![CustomerLastName]
         ![MoneyOwed] = MyTrans![MoneyOwed]
         ![CreditLimit] = MyTrans![CreditLimit]
         .UPDATE
       Else ' can only = c or d when equal
         If MyTrans![TransactionType] <> "d" Then Debug.Print "Error"
       End If
       NeedCust = 1
       NeedTrans = 1
    End If
    If Not MyLak.EOF And NeedCust Then
           MyLak.MoveNext
           NeedCust = 0
           If Not MyLak.EOF Then
              Custnum = MyLak![CustomerID]
           Else
              Custnum = 9999
           End If
     End If
     If Not MyTrans.EOF And NeedTrans Then
           MyTrans.MoveNext
           NeedTrans = 0
           If Not MyTrans.EOF Then
              Transnum = MyTrans![CustomerID]
           Else
              Transnum = 9999
           End If
      End If

Loop
MyTrans.Close        ' closing the tables
MyLak.Close
.Close
End With
End Function
```

בדוגמה זו נעבוד עם 3 טבלאות, כדלהלן:

- טבלת תנועות - טבלה שמחזיקה 3 סוגי תנועות: הוספות ("a"), מחיקות ("d") ושינויים ("c"). טבלה זו אינה מסודרת לפי מספר לקוח. כדי לבצע עדכון סדרתי, טבלת התנועות וגם טבלת הלקוחות חייבות להיות מסודרות לפי מספר לקוח. לכן, נפתח טבלה זו כקבוצה דינמית, ונקבע את המאפיין Sort כמספר הלקוח.
- טבלת לקוחות - קבוצת רשומות מסוג טבלה.
- טבלת לקוחות חדשה - בעדכון סדרתי לא מתקנים את הטבלה הישנה, אלא בונים טבלה חדשה עם התיקונים הדרושים.

עוברים על הטבלאות בעזרת פקודת MoveNext שעובדת על כל סוגי קבוצות רשומות. בקבוצות מסוג טבלה מתקדמים לפי האינדקס הפעיל, ובקבוצות דינמיות לפי סדר החזקת השאילתה.

שורה A: בניית טבלת לקוחות חדשה על ידי קריאה לשיגרה.

שורה B: קביעת האינדקס עבור קבוצת הרשומות מסוג טבלה, טבלת הלקוחות.

שורה C: קביעת סדר המיון עבור קבוצת הרשומות מסוג דינמי, טבלת התנועות.

שורות D,E: מעבר לרשומה הראשונה. שלא כמו פקודת FindFirst, פקודה זו אינה מאפשרת תנאים, אלא מעבירה אותנו לרשומה הראשונה לפי הסדר השוטף.

בשאר התוכנית נעבור דרך שתי הטבלאות המסודרות לפי מספר הלקוח. בכל פעם שרשומת התנועות קטנה מרשומת הלקוחות, נוסיף את הלקוח ברשומה זו לטבלה החדשה. כאשר רשומת התנועות גדולה מרשומת הלקוחות נעתיק את רשומת הלקוחות לטבלה החדשה. בכל פעם שרשומת התנועות שווה לרשומת הלקוחות, נבצע את התיקונים הדרושים, ונעתיק את הרשומה המתוקנת לטבלה החדשה.



תרגיל 43:

לבית הספר מצטרפים תלמידים חדשים, אחרים עוזבים, ולפעמים פרטי תלמידים משתנים, למשל כשהמשפחה גדלה. כתוב שיגרה שתקלוט את סוג הפעולה (מ' עבור מחיקה, ה' עבור הוספה, וש' עבור שינוי), מספר הזהות של התלמיד, ופרטים נוספים, אם מדובר על הוספה או שינוי.

תרגיל 44:

הכן טבלת תנועות עבור הדוגמה הקודמת, ובצע עדכון סדרתי.

תרגיל 45:

כתוב שיגרה שמאפשרת הוספת רשומות ל**טבלת השאלות מתוקנת.** בדוק שאין שגיאה כאשר נרשמים שם ומספר השואל, כלומר, לאמת על ידי בדיקה בטבלת סטודנטים שהשם שנרשם מתאים למספר שנרשם.

פעולות על תחומים

יש פעולות המיושמות על קבוצות רשומות הנקראות **תחומים** (Domains). בסיס התחום הוא טבלה או שאילתה, אך בתוך הפקודה, אפשר לעדן יותר את התחום. פעולות אלו מבוצעות בעזרת פונקציות תחומים שצורתן הכללית היא כדלהלן:

```
function name (expression, domain [,criteria])
```

הסבר: function name - אחת מהפונקציות דלהלן:

- DSum
- DCount
- DFirst
- DstDev
- DAvg
- DLookUp
- DVar

פונקציות אלו יוסברו בליווי דוגמאות.

Expression - השדה או שילוב שדות (ביטוי) שעליו נרצה להפעיל את הפונקציה.

domain - טבלה או שאילתה ממנה נשאב נתונים, ועליה נרצה להפעיל את הפונקציה.

criteria - התנאי שלפיו נרצה שתפעל הפונקציה.

תוכנית A34: דוגמה לפעולות על תחומים.

```
Function A34()
Dim FirstOne, LastOne, Phone As String, Phone0308 As Integer, _
   OwedMoney As Currency
Rem Note that the table name as well as the fieldname
Rem don't have to be surrounded by brackets if they don't
Rem contain a space.
A: OwedMoney = DCount("[MoneyOwed]", "Customers", "City='ירושלים'")
MsgBox "The total number in ירושלים is  " & OwedMoney
B: OwedMoney = DSum("[MoneyOwed]", "Customers", "City='ירושלים'")
MsgBox "the total amount owed for ירושלים is" & "  " & OwedMoney
C: OwedMoney = DSum("[MoneyOwed]", "Customers")
MsgBox "the total amount owed is" & "  " & OwedMoney
D: OwedMoney = DMin("[MoneyOwed]", "Customers", "City='ירושלים'")
MsgBox "the minimum amount owed for ירושלים is" & "  " & OwedMoney
E: OwedMoney = DMax("[MoneyOwed]", "Customers", "City='ירושלים'")
MsgBox "the maximum amount owed for ירושלים is" & "  " & OwedMoney
F: OwedMoney = DAvg("[MoneyOwed]", "Customers", "City='ירושלים'")
MsgBox "the average amount owed for ירושלים is" & "  " & OwedMoney
G: OwedMoney = DStDev("[MoneyOwed]", "Customers", "City='ירושלים'")
MsgBox "the standard deviation of amount owed for ירושלים is" & "  " & OwedMoney
H: OwedMoney = DStDevP("[MoneyOwed]", "Customers", "City='ירושלים'")
 MsgBox "the population standard deviation of amount owed for ירושלים is" & "  " & OwedMoney
Rem Also Dvar and DvarP for variance is available
I: FirstOne = DFirst("CustomerLastName", "Customers")
```

```
J: LastOne = DLast("[CustomerLastName]", "Customers")
MsgBox "This is the first customer" & "  " & FirstOne
MsgBox "This is the last customer" & "  " & LastOne
K: LastOne = DLast("CustomerLastName", "Customers", "City='ירושלים' or [City]='תל אביב'")
MsgBox "this is the last customer in ירושלים or תל אביב " & "  " & LastOne
L: Phone = DLookup("[PhoneNumber]", "Customers", "CustomerLastName='נחמני'")
MsgBox "This is the phone number of נחמני" & "  " & Phone
M: Phone0308 = DCount("[PhoneNumber]", "Customers", "[PhoneNumber]>='(03)'and
[PhoneNumber]<='(08)'")
MsgBox "The number of 03-08 area codes is" & "  " & Phone0308
End Function
```

שורה A : DCount סופרת את מספר מופעי הביטוי, במקרה זה השדה [MoneyOwed] מטבלת Customers, כאשר העיר היא ירושלים. אם הביטוי מכיל NULL ברשומה מסוימת, הרשומה אינה נכללת בספירה. אם נכתוב * במקום ביטוי, כל הרשומות תיכללנה, גם אם יש NULL באחד משדות הרשומה.

שורה B : הפונקציה DSum מחשבת את סכום הביטוי במסגרת התחום. בשורה זו נחשב את סכום חובות הלקוחות מירושלים.

שורה C : נחשב את סכום חובות כל הלקוחות.

שורה D : הפונקציה DMin מחשבת את הערך המינימלי של הביטוי במסגרת התחום. בשורה זו נחשב את החוב המינימלי של הלקוחות מירושלים.

שורה E : הפונקציה DMax מחשבת את הערך המקסימלי של הביטוי במסגרת התחום. בשורה זו נחשב את החוב המקסימלי של הלקוחות מירושלים.

שורה F : הפונקציה DAvg מחשבת את הממוצע של הביטוי במסגרת התחום. בשורה זו נחשב את ממוצע החוב של הלקוחות מירושלים.

שורה G : הפונקציה DStDev מחשבת את סטיית התקן של הביטוי במסגרת התחום. ההנחה היא שבנינו מדגם של האוכלוסיה שעבורה נרצה לחשב את סטיית התקן. בשורה זו נחשב את סטיית התקן של חוב הלקוחות מירושלים.

שורה H : הפונקציה DStDevP מחשבת את סטיית התקן של הביטוי במסגרת התחום. ההנחה היא שכללנו בחישובים את נתוני כל האוכלוסיה שעבורה נרצה לחשב את סטיית התקן. בשורה זו נחשב את סטיית התקן של חוב הלקוחות מירושלים.

שורה I : הפונקציה DFirst מאתרת את המופע הראשון של הביטוי בתחום המוגדר. בשורה זו נאתר את שם משפחת הלקוח הראשון בטבלה.

שורה J : הפונקציה DLast מאתרת את המופע האחרון של הביטוי בתחום המוגדר. בשורה זו נאתר את שם משפחת הלקוח האחרון בטבלה.

שורה K : שימוש בפונקציה DLast לאיתור שם משפחת הלקוח האחרון בטבלה שהוא או מירושלים או מתל אביב.

שורה L: הפונקציה DLookUp מחזירה את ערך הביטוי כאשר מתקיים התנאי הנקוב. התנאי אמור לקבוע רשומה ספציפית, ואם כן, ערך הביטוי הוא חד-משמעי. אם יש מספר רשומות המקיימות את התנאי, נחזיר את ערך הביטוי ברשומה הראשונה שמקיימת אותו, ואם אין רשומה המקיימת את התנאי, נחזיר ערך Null. בשורה זו נאתר מספר טלפון של לקוח בשם נחמני.

שורה M: נאתר את מספר הלקוחות שקידומת מספר טלפון שלהם היא בין 03 ו-08.



תרגיל 46:

כתוב שיגרה המבררת מה מספר העובדים הירושלמיים.

תרגיל 47:

כתוב שיגרה המבררת מה מספר העובדים הזכרים שגרים בירושלים.

תרגיל 48:

כתוב שיגרה המבררת את ממוצע מבחני הכניסה של העובדים.

תרגיל 49:

כתוב שיגרה לאיתור ממוצע מבחני הכניסה של העובדים הירושלמיים.

תרגיל 50:

כתוב שיגרה המבררת את הציון המינימלי במבחני הכניסה לעובדים.

תרגיל 51:

כתוב שיגרה לאיתור הציון המינימלי במבחני הכניסה לעובדים זכרים.

תרגיל 52:

כתוב שיגרה לאיתור הציון הממוצע של תלמיד שאת שמו נקבל כקלט.

תרגיל 53:

כתוב שיגרה המאתרת ממוצע וסטיית תקן של מספר הילדים במשפחות תלמידי בית הספר.

תרגיל 54:

כתוב שיגרה המחשבת את סכום מספר הילדים במשפחות כל התלמידים בבית הספר.

פרק 6

ניפוי תוכניות

מבוא

מטרת **הניפוי** (Debugging) היא למצוא את מקור הבעיה כאשר תוכנית אינה מחזירה את התוצאה הרצויה. כלי ניפוי ב-Access מאפשרים לנו לעשות את הדברים הבאים:

- לעבור דרך התוכנית צעד צעד, ולבדוק בכל שלב את מצב התוכנית.
- לקבוע **נקודות עצירה** (BreakPoints) כדי לעצור את התוכניות בנקודות מסוימות, או כאשר ערך משתנה כלשהו משתנה.
- לעקוב אחרי שינויים בערך של כל משתנה במשך התוכנית בחלון הניפוי.
- להקליד פקודות ישירות לחלון הניפוי, כדי לבדוק היבטים מעניינים של סביבת התוכנית, להריץ שגרות, וכולי.

קביעת נקודות עצירה

פתח את המודול שנקרא Messages, והבט בפונקציה Confirm.

תוכנית Confirm:

```
Public Function Confirm(strMessage As String) As Boolean
1: Dim bytChoice As Byte
2: bytChoice = MsgBox(strMessage, vbQuestion + vbOKCancel, conAppName)
3: If bytChoice = vbOK Then
4:    Confirm = True
5:  Else
6:    Confirm = False
7: End If

8: End Function
```

הפונקציה מציגה הודעה מסוימת בתיבת הודעה (MsgBox), ואם המשתמש לחץ על אישור היא מחזירה את הערך True, אחרת היא מחזירה את הערך False. בצע את הצעדים הבאים:

1. קבע נקודת עצירה בשורת הכותרת של הפונקציה על ידי לחיצת הלחצן המתאים בסרגל הכלים, מתפריט **אתר באגים** (Debug) בחר **החלף מצב נקודת עצירה** (Toggle BreakPoint), או הקש F9 כאשר הסמן נמצא בשורה הראשונה.
2. פתח את החלון לאיתור באגים על ידי לחיצת הלחצן המתאים בסרגל הכלים, בחירתו מתפריט **תצוגה**, או הקש G^.
3. בחלק התחתון של החלון הקלד:

```
? Confirm ("2 and 2 are 4")
```

והקש Enter. שים לב שהפונקציה עוצרת בשורה 1 שמסומנת בצהוב ובחץ.

4. התקדם צעד צעד בלחיצה על הלחצן **צעד לתוך** (Step Into) שבסרגל הכלים. התקדם לפקודה 1 אך על תעשה דבר, משום שפקודה 1 אינה פקודה ביצועית, רק פקודת הצהרה. המשך לפקודה 2 ולחץ שוב על הלחצן, ההודעה תופיע על המסך. לחץ על **אישור**, והשיגרה תתקדם לפקודה 3. בפקודה 3 לאחר הלחיצה הבאה על **צעד לתוך**, תתקדם לפקודה 4 (משום שהתנאי בפקודת IF מתקיים). לחץ שוב ותתקדם לפקודה 5, אך בלחיצה הבאה תגיע לפקודה 7. המנפה דילג על פקודה 6, משום שהתנאי אינו מתקיים ולכן אין צורך לבצע אותה.
5. המשך ללחוץ על הלחצן עד לגמר הפונקציה. בנקודה זו חלון איתור הבאגים חוזר ומופיע. אך גם לפני סיום ביצוע הפונקציה אפשר להפעילו בכל עת. בחלק העליון של החלון תוכל לראות את הערכים הנוכחיים של כל המשתנים המקומיים המופיעים בפונקציה.
6. סגור את החלון והורד את נקודת העצירה על ידי בחירת **נקה את כל נקודות העצירה** (Clear All BreakPoints) מתפריט **אתר באגים**.

מעבר משיגרה לשיגרה: השיגרה Addresses קוראת לפונקציה Confirm.

תוכנית Addresses:

```
Public Sub Addresses(Address As String, Optional Zip As Variant)
1: Dim booOK As Boolean
2: If IsMissing(Zip) Then
3:     booOK = Confirm("You forgot the Zip. Alright?")
4:  If Not booOK Then
5:      Zip = InputBox("Enter Zip value")
6:   End If
7: End If
End Sub
```

נקבע נקודת עצירה בשורה 3. נריץ את השיגרה מחלון איתור באגים על ידי הקלדת הפקודה:

```
Call Addresses ("111 Herzl")
```

השיגרה תרוץ ותעצור בשורה 3. בנקודה זו אפשר ללחוץ על הלחצן **צעד לתוך** ולהיכנס לפונקציה Confirm, או ללחוץ על **צעד מעל** (Step Over) ולבצע את הפונקציה Confirm (אך לא לצעוד דרכה) ולהמשיך עם הפקודה הבאה שבשיגרה Addresses.

נניח שבחרנו **צעד לתוך**. במקום להמשיך צעד צעד, עבור לשורה 8 בפונקציה, לחץ לחיצה ימנית ובחר **הפעל עד לסמן** (Step To Cursor). בדרך זו יש אפשרות לדלג על מספר פקודות. כאשר השורה האחרונה הופכת צהובה, אפשר להמשיך עם **צעד לתוך**, עד לסיום השיגרה.

צפייה במשתנים

אפשר להפעיל צפייה על כל משתנה או שדה בטבלה. צפייה מהירה במשתנה מציגה את ערכו הנוכחי. מקם את הסמן בשורה 1 של השיגרה Addresses, על משתנה booOK, ולחץ על הלחצן **צפייה מהירה** (Instant Watch) שבסרגל הכלים, או בחר **צפייה מהירה** מתפריט **אתר באגים**. שם המשתנה יופיע בחלון, וכאשר תצעד דרך התוכנית, ערך המשתנה יעודכן (ישתנה) בהתאם לשלבים שבתוכנית. כדי למוחקו, בחר בו בחלון איתור באגים והקש Del.

נציג דוגמה שנייה על ידי ביצוע הפונקציה Watch. נתקין נקודת עצירה על הפקודה Debug.print I, ונגדיר צפייה מהירה במשתנה I. נוכל להפעיל את הפונקציה על ידי לחיצה על הלחצן **לך** (F5), ואחר על הלחצן **חלון איתור באגים** (^G) כדי לראות את הערך המעודכן של I. אפשר להמשיך כך בהקשת F5 ו-^G לסירוגין, ובכל פעם נבדוק את הערך המעודכן של I בחלון, עד לסוף הביצוע.

פרק 7

חיבור לטפסים

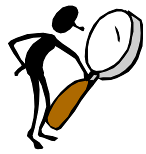
בניית מסד נתונים

לפני שנשתמש בקוד כדי לשפר מסד נתונים, יש צורך במסד נתונים. נשתמש באשף מסדי נתונים כדי לבנות אותו. פתח את Access ובחר **אשף מסדי הנתונים**. בחר את מסד הנתונים בשם Order Entry, ולחץ על **אישור**, מקם אותו במקום רצוי ובחר **צור** (Create). לחץ על **הבא**, בחר לכלול מדגם של נתונים, ולחץ שוב על **הבא**. בחר צבע, סגנון ושם ולחץ על **סיום**.

בניית לחצן להוספת רשומות

בטופס שמוצג ברצוננו להוסיף לחצן שיאפשר להוסיף רשומה באופן מיידי, מבלי לגלול עד לסוף הנתונים. בצע את הצעדים הבאים:

1. ממסך הניווט (Switchboard) שיוצג בחר **הזן/הצג מידע אחר**, ואחר בחר **הזן/הצג עובדים**.
2. עבור למבט **עיצוב** על ידי בחירת **תצוגת עיצוב** מתפריט **תצוגה**.
3. בארגז הכלים, כאשר אשפי הבקרה מופעלים, לחץ על **לחצן פקודה**, וצייר לחצן על הטופס.
4. במסך הראשון של האשף, בחר בצד ימין פעולות רשומה (Record Operations), ובשמאל בחר **הוסף רשומה חדשה** (Add New Record) ולחץ **הבא**.
5. בחר תמונה ולחץ על **הבא**.
6. בחר את השם **AddRecord** עבור הלחצן, ולחץ **סיום**.
7. סגור את הטופס על ידי לחיצה על לחצן שמאל (X בפינה הימנית העליונה) ושמור את השינוי.



הערה:

אם בשלב כלשהו תצא מהיישום וכשתחזור אליו מסך הניווט לא יוצג, עבור לכרטיסיה **טפסים**, בחר **SwitchBoard** ולחץ על **פתיחה**. בחר שוב **הזן/הצג מידע אחר**, ואחר בחר **הזן/הצג עובדים**. בטופס שיוצג תראה את הלחצן החדש ותוכל להפעילו.

שיפור הלחצן להוספת רשומות

כאשר נוסיף רשומה חדשה בעזרת הלחצן שבנינו, הסמן לא יעבור לשדה הראשון ויש צורך למקמו ידנית. השדה הרצוי הוא שם פרטי. בתצוגת עיצוב של הטופס שנה את שם הפקד שם_פרטי ל-FirstName (קל יותר לעבוד עם שמות באנגלית). עבור ללחצן AddRecord. לחץ לחיצה ימנית ובחר **אירוע בנייה** (Build Event). לאחר פקודה 1 הוסף את פקודה 2:

```
1: DoCmd.GoToRecord     , , acRecordNew
2: FirstName.SetFocus
```

הסבר: הפקודה הראשונה שייכת לאובייקט DoCmd, והיא נכתבה על ידי האשף כדי להפנות אותנו לרשומה חדשה (למשמעות הפרמטר acRecordNew, עיין בעזרה).

הפקודה השנייה מעבירה את המיקוד לפקד שנקרא כעת FirstName.

שינוי מצב טופס עובדים ל"קריאה בלבד"

במבט עיצוב של הטופס **עובדים** לחץ פעמיים על ריבוע בחירת הטופס בפינה השמאלית עליונה של הטופס כדי להציג את מאפייני הטופס. עבור לכרטיסיה **נתונים** ובמאפיין **אפשר עריכות** (AllowEdits) בחר **לא**. כעת אי אפשר לערוך את הטופס.

אך אנו רוצים לאפשר ביצוע עריכה על ידי לחיצה על לחצן. בארגז הכלים כבה את הלחצן **אשפי בקרה**, בחר **לחצן פקודה** ובנה לחצן. סמן אותו, לחץ לחיצה ימנית ובחר **מאפיינים**. במאפיין שם הקלד EditRecord, ובכיתוב הקלד **&ערוך רשומה**, המאפשר לנו להקיש על **Alt+ע** במקום ללחוץ על הלחצן.

כדי שהלחצן יאפשר עריכה, לחץ שוב לחיצה ימנית, בחר **אירוע בנייה**, ואחר **בונה קוד**. בחלון הקוד שיופיע הקלד:

```
Me.AllowEdits = True
FirstName.SetFocus
```

הסבר: השורה הראשונה מאפשרת עריכה, כלומר היא משנה את ברירת המחדל שקבענו, עד למעבר לרשומה חדשה.

בניית לחצן לשמירת רשומות

ברצוננו להוסיף לחצן שיאפשר לנו לשמור רשומה באופן מיידי, מבלי לעבור לרשומה הבאה או לסגור את הטופס. נבצע את הצעדים הבאים:

1. מטופס הניווט בחר הזן/הצג מידע אחר, ואחר הזן/הצג עובדים.
2. עבור למבט **עיצוב**.
3. בארגז הכלים, כאשר האשף מופעל לחץ על **לחצן פקודה** וצייר לחצן על הטופס.
4. במסך הראשון של האשף, בחר בצד ימין **פעולות רשומה** (Record Operations), ובצד שמאל **שמור רשומה** (Save Record) ולחץ **הבא**.
5. בחר טקסט ואת הכיתוב **שמור רשומה** ולחץ על **הבא**.
6. בחר את השם SaveRecord ולחץ **סיום**.
7. סגור את הטופס על ידי לחיצה על לחצן שמאל (X בפינה הימנית העליונה) ושמור את השינוי.

ביטול אפשרות העריכה

ברצוננו לבטל את אפשרות העריכה בעת עדכון הרשומה הנוכחית או בעת מעבר לרשומה אחרת.

1. במבט עיצוב לחץ פעמיים על הריבוע השחור בפינה השמאלית העליונה של הטופס כדי לעבור למאפייני הטופס.
2. לחץ על הכרטיסיה **אירוע** (Event), בחר **בנוכחי** (OnCurrent), בחר בבונה (**...**), ובחר **בונה קוד**.
3. הקלד: Me.AllowEdits=False.
4. העתק את הקוד שהקלדת בדרך המקובלת (C^).
5. מתפריט **חלון**, בחר **עובדים:טופס**.
6. במאפיין **לאחר עדכון** (AfterUpdate) לחץ על הבונה, בחר **בונה קוד**, ולחץ **אישור**.
7. הקש V^ כדי שהקוד שהקלדת קודם יועתק למאפיין **לאחר עדכון**.
8. הוסף עוד פקודה: ".MsgBox "Record Saved.

תיבה משולבת לאיתור רשומה מסוימת

ברצוננו להוסיף לחצן שיאפשר לנו לאתר רשומה כלשהי באופן מיידי, מבלי לסרוק את הטבלה בצורה סדרתית. נבצע את הצעדים הבאים:

1. מטופס הניווט בחר **הזן/הצג מידע אחר**, ואחר **הזן/הצג עובדים**.
2. עבור למבט עיצוב, מארגז הכלים, כאשר האשף מופעל, בחר תיבה משולבת.
3. בחר באפשרות השלישית (**מצא רשומה בטופס שלי המבוססת על הערך אותו בחרתי בתיבה המשולבת שלי**). לחץ על **הבא**.
4. כאשר מוצגת השאלה איזה שדה להציג, בחר **שם מוצר**, ולחץ על **הבא**.
5. קרא לתיבה **שם מוצר**.

ניתן לשפר את השימוש בתיבה המשולבת במספר דרכים:

- להדגיש ולצבוע אותה בצבע השונה מיתר השדות המיועדים לקלט.
- לבחור במאפייני התיבה המשולבת, ללחוץ על הכרטיסיה **נתונים**, ללחוץ על הבונה ליד **מקור שורה**, ולהקליד בטור שכתוב בו **שם_מוצר** את הביטוי:

```

```

ביטוי זה יאפשר לנו לראות בתיבה המשולבת את שם המוצר וגם את מספרו.

כאשר גוללים את הרשומה בעזרת החיצים שבתחתית הטופס, שים לב שהתיבה המשולבת אינה מתעדכנת. הבעיה נוצרה משום ששם התיבה המשולבת הוא משולבת6. במאפייני הטופס כולו לחץ על הכרטיסיה **אירוע**, במאפיין **בנוכחי** לחץ על הבונה, בחר **בונה קוד**, ומלא את הקוד דלהלן:

```
משולבת6 = [קוד_מוצר]
```

כעת, בכל פעם שקוד המוצר ישתנה (גם אם עברת לרשומה הבאה) תוכן התיבה המשולבת ישתנה אף הוא.

צביעת התיבה בצבע שונה מדגיש את ההבדל בינה לבין השדות האחרים. אך כאשר נגיע לשדה זה נרצה שיהפוך ללבן כמו כל שדה רגיל. נבנה קטע קוד שמטרתו לשנות את הצבע של התיבה ללבן, ונדביקו לאירוע **בעת הזנה** (On Enter).

נקליד את הפקודה הבאה בתוך השיגרה:

```
משולבת6.BackColor = 16777215
```

כדי שהתיבה תחזור לצבעה כאשר יוצאים ממנה, נדביק שורה מקבילה לאירוע בעת יציאה (On Exit):

```
משולבת6.BackColor = 12632256
```

סינון לפי טופס

כדי לבצע סינון לפי טופס בטופס מוצרים, לחץ על הלחצן המתאים בזמן שהטופס פתוח. כדי להציג הודעה מלווה בהסברים לשימוש בסינון לפי טופס עבור למבט עיצוב, לחץ על הלחצן **קוד**, כאשר החלון נפתח, בחר **Form** ברשימה הנפתחת משמאל למעלה ובחר **Filter** ברשימה הנפתחת בצד ימין למעלה. בשיגרה שתיפתח הקלד:

```
If filterType = acFilterByForm Then
        MsgBox "בחר את ערכי השדות. אחר כך, לחץ על הלחצן החל מסנן."
End If
```

כעת, בכל פעם שנפעיל את המסנן נקבל הודעה זו. פקודת IF נחוצה כדי שההודעה לא תוצג בכל סוג סינון, רק ב**סינון לפי טופס**.

קבוצת אפשרויות לסינון נתונים

נוסיף את השדה City לטבלת עובדים. בטופס עובדים נבנה קבוצת אפשרויות בת שתי אפשרויות, הראשונה - All Cities (תקבל ערך 1, ותהיה ברירת המחדל), השנייה תהיה New York (תקבל ערך 2). נשמור את התוצאה לשימוש מאוחר יותר. במאפייני קבוצת האפשרויות נקבע את שם הפקד כ: FilterOptions.

באירוע AfterUpdate של קבוצת האפשרויות נכתוב את הקוד הבא:

```
If FilterOptions = 2  Then
Me.Filter = "City = 'New York' "
Me.FilterOn = True
Else
      Me.FilterOn = False
End If
```

הסבר: כאשר נבחרת אחת האפשרויות המשתנה FilterOptions מקבל אחד מהערכים 1 או 2. אם הערך הוא 2, נגדיר במסנן שהשדה City יקבל את ערך של New York, אחרת נכבה את המסנן. המילה Me שווה לביטוי: Forms!FormName, כאשר הטופס הוא הטופס שהקוד קשור אליו. המאפיין Filter מכיל את הגדרת המסנן, כאשר FilterOn מפעיל את המסנן כשערכו שווה True.

עדיין קיימת בעיה. אם מפעילים מסנן אחר, קבוצת האפשרויות אינה משתנה וזה מטעה. למשל, אם הפעלנו סינון לפי העיר Cleveland, תיבת האפשרות All Cities תישאר מסומנת. כדי לשפר, נכתוב את הקוד הבא באירוע OnFilter של הטופס.

```
FilterOptions = Null
```

הסבר: שיגרה זו מופעלת כאשר מפעילים סינון לפי טופס. במקרה זה איננו רוצים שהאפשרויות תופענה, מפני שסינון לפי טופס יכול להיות שונה מאחרים.

קביעת ערכים כתוצאה מעדכון שדות מסוימים בטבלה

נוסיף לטבלת עובדים את השדה **שם למכתב**, שמשמעותו היא כינוי שמשתמשים בו בפתיח מכתב, ונוסיף אותו לטופס. עד להזנת השדה נרצה שיקבל את הערך שנמצא בשדה **שם_פרטי**. באירוע לאחר עדכון (After Update) של השדה **שם_פרטי** נרשום את הפקודה הבאה:

```
If IsNull ([שם למכתב]) Then
	[שם למכתב] = [שם_פרטי]
End If
```

אם השדה **שם למכתב** ריק, הפקודה מעתיקה לשדה זה את תוכן השדה **שם_פרטי**.

באותה דרך יכולנו להזין את העיר כתוצאה מהזנת הקוד האזורי, על ידי הוספת הקוד הבא לאירוע **לאחר עדכון** של השדה ZIP:

```
If ZIP >= "44100" and ZIP <= "44199"  Then
	City = "Cleveland"
End If
```

הפעלת שגרות באמצעות צירוף מקשים

נניח שרוב העובדים גרים ב- New York. במקום להזין שוב אותו ערך נרצה לאפשר קיצור דרך, למשל אם נקיש Ctrl+1 הערך New York ייכנס באופן אוטומטי, מבלי שנצטרך להקלידו. נכתוב את הקוד הבא באירוע **בעת ירידת מקש** (KeyDown) של השדה City:

```
If KeyCode = vbKey1 and Shift = acCtrlMask Then
  City = "New York"
End If
```

הקבוע vbKey1 מציין את קוד ASCII של הספרה 1, והקבוע acCtrlMask מצביע על המקש Control. קוד זה מאפשר שימוש בצירוף מקשים רק כאשר הסמן נמצא בשדה City. ומה אם נרצה שיפעל מכל מקום? ניתן לסדר זאת על ידי ביצוע שני תיקונים קלים:

- באירועי הטופס כולו קבע **מקש תצוגה מקדימה** (KeyPreview) **כן**. שינוי זה גורם לכך שלפני ביצוע השגרות הקשורות לאירועי שדה כלשהו, תהיה התייחסות לאירועי הטופס כולו.
- במקום להדביק את השיגרה הקודמת לשדה City, הדבק אותה לאירוע **בעת ירידת מקש** של הטופס.

אימות נתונים

נניח שחובה להזין ערך בשדה המיקוד רק אם השדה CITY נכלל. במקרה כזה, נדביק את הפקודות האלו לשיגרה **לפני עדכון** של הטופס כולו:

```
Dim strMessage As String
Dim intOptions As Integer
Dim bytChoice As Byte
If Not IsNull(CITY) And IsNull(ZIP) Then
  strMessage = "You didn't enter ZIP. Save anyway"?
  intOptions = vbQuestion + vbOKCancel
  bytChoice = MsgBox(strMessage, intOptions)
  If bytChoice = vbCancel Then
        ZIP.SetFocus
        Cancel = True
  End If
End If
```

הסבר: כדי לבטל שמירת רשומה כאשר המיקוד אינו מוזן נחזיר את הסמן לשדה ZIP.

הוספת ערכים לתיבה משולבת

נוסיף טבלה **מדינות** ובה השדות מספר אוטומטי ושם. נוסיף שדה STATE לטבלת עובדים, תיבה משולבת לטופס עובדים שתאפשר להזין שם מדינה, ומספרה ייכנס לטבלת עובדים (להוספת שדות ופקדים ראה פרק 6 בספר **Access 97 סדנת לימוד** מאת פרופ' אבא אנגלברג). באירוע התיבה **בעת שלא ברשימה** (NotInList) הקלד:

```
Private Sub משולבת45_NotInList(NewData As String, Response As Integer)
Dim strMessage As String
Dim dbsOrders As Database
Dim rstStates As Recordset
strMessage = "Are you sure you want to add  "' & NewData & "' to the list of States?"
If Confirm(strMessage) Then
  Set dbsOrders = CurrentDb()
  Set rstStates = dbsOrders.OpenRecordset("מדינות")
  rstStates.AddNew
  rstStates![שם מדינה] = NewData
  rstStates.Update
  Response = acDataErrAdded
Else
  Response = acDataErrDisplay
End If
End Sub
```

בשיגרה זו נוסיף את שם המדינה שהוזן לטבלת **מדינות**. למזלנו, מספר המדינה הוא מסוג מספור אוטומטי, ולכן איננו צריכים להוסיף אותו. אם מספר המדינה היה מסוג מספר רגיל, היינו צריכים להציג InputBox בשיגרה כדי לקלוט אותו, או אם קיימים פרטים רבים להוספה, להציג טופס שלם לקליטת כל נתוני המדינה החדשה. את השיגרה Confirm, נקליד במודול שנקרא **קוד כללי**, והוא ייראה כך:

```
Public Function Confirm(strMessage As String) As Boolean
Dim bytChoice As Byte
bytChoice = MsgBox(strMessage, vbQuestion + vbOKCancel, conAppName)
If bytChoice = vbOK Then
   Confirm = True
Else
   Confirm = False
End If
End Function
```

איתור רשומה רצויה בעזרת תיבת דו-שיח

למדנו כיצד לאתר רשומה מסוימת בעזרת תיבה משולבת מהסוג השלישי. דרך אחרת היא לבנות טופס דו-שיח שמחזיק פרטים מזהים של רשומות הטבלה. לאחר זיהוי הרשומה, היא מוצגת בטופס.

1. בנה טופס ריק וקרא לו Opener. בארגז הכלים, כאשר האשף מופעל בחר תיבת רשימה (List Box). במסך שמופיע, בחר: **ברצוני שתיבת רשימה תברור את הערכים מטבלה או משאילתא**.
2. לחץ **הבא** ובחר טבלת **עובדים**. לחץ **הבא** ובחר את השדות (שם_משפחה, שם_פרטי, City, ו-ZIP).
3. לחץ **הבא** והתאם את גודל העמודות על ידי לחיצה כפולה בראש כל עמודה.
4. לחץ **הבא** וקבע את תווית תיבת הרשימה כ**בחר עובד** ולחץ **סיום**.
5. מקם את התווית במקום הרצוי מעל תיבת הרשימה. כאשר הרשימה מוצגת, שים לב שסדר הרשומות הוא כפי שהוא בטבלה, כלומר לפי מספר הזהות שהוא המפתח הראשי. כדי שיופיעו בסדר אלפביתי, עבור למאפייני תיבת הרשימה, בכרטיסיה **נתונים**, במאפיין **שורת מקור** לחץ על הבונה. בתצוגת השאילתה שתופיע, בעמודת שם_משפחה ובשורה **מיין**, לחץ על הלחצן וקבע **סדר עולה**.

הטופס ותיבת הרשימה מהווים תיבת דו-שיח (Dialog Box), כלומר תיבה שמטרתה לאפשר המשך חלק של שאר התוכנית.

במצב הנוכחי יש בטופס מספר תכונות שאינן מקובלות בתיבת דו-שיח טיפוסית, למשל לחצני ניווט ופסי גלילה. כדי להיפטר מסממנים אלה, בצע את הפעולות הבאות.

עבור למאפייני הטופס כולו, בכרטיסיה **תבנית** מלא את הערכים הבאים:

- כיתוב (Caption): עבור לעובד
- פסי גלילה (Scroll Bars): ללא
- בוררי רשומות (Record Selectors): לא
- לחצני ניווט (Navigation Buttons): לא
- מרכוז אוטומטי (AutoCenter): כן
- סגנון גבול: דו-שיח

כעת עבור לכרטיסיה **אחר**

- קבע את המאפיין **מודאלי** ל**כן**. בחירה זו תמנע את האפשרות של לחיצה בכל חלון מחוץ לתיבת הדו-שיח.

כעת נרצה ליצור מצב שכאשר המשתמש בוחר רשומה ומקיש Enter, או לוחץ לחיצה כפולה על כניסה כלשהי, הרשומה תוצג אוטומטית.

נוסיף שתי שגרות באירועי תיבת הרשימה. באירוע **בעת לחיצה כפולה** נקליד את השיגרה הבאה:

```
DoCmd.OpenForm "עובדים",,,"קוד עובד = " & רשימה0
```

בעזרת פקודה זו אנו פותחים את טופס עובדים כשהרשומה המוצגת היא זו שערך שדה המפתח שלה: **קוד עובד** שוה לערך הנוכחי של תיבת הרשימה: **רשימה0.**

כדי שכך יקרה גם בעת הקשת ENTER, נדביק לאירוע **בעת לחיצת מקש** (On KeyPress) את השיגרה הבאה:

```
If KeyAscii = vbKeyReturn Then
  DoCmd.OpenForm "עובדים",,,"קוד עובד = " & רשימה0
End If
```

הסבר: אנו רוצים לעבור לטופס רק כאשר הקשנו על Enter, ולא על כל מקש אחר, ולכן נחוצה פקודת IF.

ברצוננו לאפשר בטופס חזרה לתיבת הדו-שיח. כדי לעשות זאת, כאשר האשף מופעל, נוסיף לחצן פקודה ומתוך **פעולות טופס** נבחר **סגור טופס.**

התקנת תיבת דו-שיח בצורה אלטרנטיבית

נניח שנרצה להשתמש בתיבת דו-שיח בצורה שונה. במקום לגלול את הרשומות עד לבחירת הרשומה הרצויה, נלחץ על לחצן בשם **עבור לרשומה רצויה**, שמייד נבנה.

העתק את טופס Opener וקרא לטופס שתדביק RecordFinder.

כדי שתוכל לעבור לטופס RecordFinder, בטופס **עובדים** כאשר האשף מופעל, צור לחצן פקודה, בחר **פעולות טופס**, **פתח טופס** וקבע הפנייה לטופס RecordFinder. הכיתוב על הלחצן יהיה : &**עבור לרשומה רצויה**.

כאשר האשף מכובה בנה בטופס RecordFinder לחצן פקודה עם הכיתוב &**מצא רשומה**, והדבק את השיגרה הבאה לאירוע **בעת לחיצה** :

```
Private Sub מצא_רשומה_Click()
Dim rst As Recordset
Set rst = Forms![עובדים].RecordsetClone
rst.FindFirst  "קוד עובד = " & רשימה0
Forms![עובדים].Bookmark = rst.Bookmark
DoCmd.Close acForm, "RecordFinder"
End Sub
```

הסבר : הפקודה RecordsetClone בונה קבוצת רשומות (נוספת) המבוססת על הטבלה שעליה מבוסס טופס עובדים. היתרון הוא שלפעמים נרצה לנוע בקבוצה זו מבלי לנוע בטבלה המקורית. בדוגמה זו, לאחר איתור הרשומה המחזיקה את קוד העובד הרצוי, שקובע את מספר הרשומה המוחזק בסימניה (rst.Bookmark), נשווה את הסימניה בטבלה המקורית לאותה רשומה. הפעולה גורמת למעבר אל רשומה זו בטופס הפתוח. בנקודה זו נסגור את הטופס RecordFinder, הטופס המקורי **עובדים** יישאר גלוי ויציג את ברשומה הנכונה.

נניח שנרצה לקבוע את הרשומה המתאימה לא רק על ידי לחיצת הלחצן **מצא רשומה**, אלא גם על ידי לחיצה כפולה על הרשומה הרצויה בתיבת הרשימה. נדביק לאירוע **בעת לחיצה כפולה** (On Double Click) את הקוד הבא :

```
Private Sub רשימה0_DblClick(Cancel As Integer)
If Not IsNull(רשימה0) Then
  מצא_רשומה_Click
End If
End Sub
```

הסבר : רישום שם השיגרה **מצא_רשומה_Click** גורם להפעלת השיגרה שתיארנו עבור קביעת הרשומה הרצויה בטופס עובדים. הבדיקה IsNull תמנע את הפעלת השיגרה אם בטעות תבוצע לחיצה כפולה על כותרת הרשימה ולא אחת מהכניסות.

בזמן בניית הלחצן **מצא רשומה**, ב**מאפיינים**, לאחר בחירה בכרטיסיה **נתונים**, קבע את הערך של **אפשר** ל**לא**. מצב זה ימנע את השימוש בלחצן עד שנבחר רשומה מסוימת מתיבת הרשימה. כאשר נקבע את הרשומה, נהפוך את המאפיין **אפשר** ל**כן** על ידי הקלדת קטע קוד זה באירוע ל**אחר עדכון** (After update) של תיבת הרשימה:

```
Private Sub 0רשימה_AfterUpdate()
מצא_רשומה.Enabled = True
End Sub
```

כדי לאפשר ביטול בחירה, כאשר האשף מופעל צור לחצן, בחר **פעולות טופס**, **סגור טופס**. כדי שהטופס ייסגר גם בעת הקשת Escape, ב**מאפיינים** בכרטיסיה **אחר**, קבע את ערך **ביטול** ל**כן**.

שימוש בתיבת דו-שיח בדוחות

בנה דוח אוטומטי שמבוסס על טבלת עובדים וייקרא **עובדים**. פתח את הדוח במבט **עיצוב**, הצג את מאפייניו ולחץ על הכרטיסיה **נתונים**. במאפיין **מסנן** (Filter) רשום: City = "Cleveland", ובמאפיין **המסנן מופעל** (FilterON) רשום **כן**. אם תציג את הדוח לפני הדפסה, תראה עובדים שגרים ב-Cleveland בלבד. בצורה זו אפשר לסנן את הנתונים בכל דרך שהיא, אך תהליך הסינון דורש מעט ידע ב-Access ולא ראוי לדרוש ידע כזה מהמשתמש. בנה תיבת דו-שיח כדי לאפשר למשתמש לבצע סינון.

העתק את הטופס Opener וקרא לטופס שיודבק Workers. הסר את תיבת הרשימה, ובנה קבוצת אפשרויות בת 3 אפשרויות: כל העובדים, עובדים מ-Cleveland ועובדים מ- New York. בתחתית הטופס מקם 3 לחצנים, **הדפס**, **הצג לפני הדפסה** ו**בטל**. את הלחצן השלישי בנה בעזרת האשף ובחר **פעולות טופס**, **סגור טופס**. הדבק לאירוע **בעת לחיצה** של הלחצן **הדפס** את הקוד הבא:

```
Dim strFilter As String
Select Case ReportType
Case 2
strFilter = "City = 'Cleveland'"
Case 3
strFilter = "City = 'New York'"
End Select
If ReportType = 1 Then
  DoCmd.OpenReport "עובדים"
Else
   DoCmd.OpenReport "עובדים", acNormal, , strFilter
End If
```

אם המשתמש בחר באפשרות הראשונה יודפס דוח רגיל. כדי לאפשר לו לבחור באפשרות השנייה או שלישית, הגדר מסנן וכלול אותו בפקודת OpenReport.

הקוד זהה גם עבור הלחצן **הצג לפני הדפסה**, אך בפקודת OpenReport במקום הפרמטר acNormal שמציג דוח רגיל, הקלד acPreview שמציג הצגה מקדימה שלו.

הצגת טבלאות קשורות

בנה טבלה חדשה הקשורה לטבלת **עובדים** שתיקרא טבלת **משכורות**. בטבלה זו השדות הבאים:

- מספר עובד
- שנה
- חודש
- שכר
- מס הכנסה

ברצוננו להציג טבלת **משכורות** יחד עם טבלת **עובדים**, כך שנוכל לראות את משכורות עובד מסוים יחד עם פרטיו. בנה טופס אוטומטי, טבלאי, עבור טבלת **משכורות** על ידי לחיצה על הלחצן **אובייקט חדש** שבסרגל הכלים. גם לטופס החדש קרא **משכורות**.

כדי שתוכל לראות את טופס **משכורות** יחד עם טופס **עובדים**, קבע את המאפיינים הבאים של טופס **משכורות** כדלהלן:

1. לחץ על **מאפיינים**, **אחר**, וקבע **מוקפץ** (Popup) **לכן**. טופס מוקפץ יופיע תמיד מעל כל הטפסים האחרים הפתוחים. אם נציג גם טופס עובדים וגם טופס משכורות, טופס משכורות יישאר מעל ולא יכוסה.
2. לחץ על **תבנית** (Format), וקבע **תצוגת ברירת מחדל** (Default Display) כ**טפסים רציפים** (Continuous).
3. בחר בכרטיסיה **נתונים** (Data), אם אין ברצונך לאפשר שינויים בטופס משכורות שיופיע עבור כל עובד, קבע **אפשר עריכות** (AllowEdits), **אפשר מחיקות** (AllowDeletions) **ואפשר תוספות** (AllowAdditions) **ללא**.
4. בנה לחצן פקודה בעזרת האשף כדי לסגור את הטופס.
5. כעת חזור לטופס עובדים. בנה לחצן פקודה בעזרת האשף ופעולות טופס כדי לפתוח את הטופס **משכורות**, כאשר הקשר בין הטפסים הוא **מספר עובד** בטבלת **משכורות**, **וקוד_עובד** בטבלת **עובדים**.
6. כדי לגרום לשינוי בתוכן טופס משכורות בעת מעבר מרשומה לרשומה, הוסף את הקוד הבא לאירוע **בנוכחי** (OnCurrent) של הטופס **עובדים**:

```
If IsOpen("משכורות") Then
  Forms![משכורות].Filter = "[מספר עובד] = " & Nz ([קוד_עובד],0)
End If
```

הסבר: השיגרה IsOpen בודקת אם הטופס **משכורות** פתוח. היא מוגדרת כך:

```
Public Function IsOpen(ByVal strFormName As String) As Boolean
IsOpen = False
If SysCmd(acSysCmdGetObjectState, acForm, strFormName) <> 0 Then
  If Forms(strFormName).CurrentView <> 0 Then
    IsOpen = True
  End If
End If
End Function
```

פונקציה SysCmd מאפשרת בדיקת מצב המערכת. הפרמטר acSysCmdObjState, קובע שאנו מנסים לברר מצב אובייקט כלשהו. הפרמטר השני acForm, מבהיר שהאובייקט שמעניין אותנו הוא טופס, והפרמטר השלישי הוא שם המחרוזת המחזיקה את שם האובייקט שעליו אנו רוצים לברר מידע. אם הפונקציה שווה ל-0, האוביקט סגור. אחרת, הוא פתוח.

אם הטופס פתוח, המאפיין Currentview שווה ל-0 אם הוא פתוח במצב עיצוב, אחרת הוא פתוח במצב קליטת נתונים, ורק אז נחזיר ערך True לפונקציה שלנו.

לאחר שנבדוק אם הטופס פתוח, נקבע במאפיין **מסנן** (Filter) של טופס **משכורות** את **מספר עובד** (שם השדה בטבלת משכורות) = **קוד_עובד** (שם השדה בטבלת עובדים). הפונקציה Nz (NullZero) הופכת את הערך של **קוד_עובד** ל-0 אם משום מה הוא Null (אחרת, ערכו לא משתנה).

כדי לאפשר עדכונים בטופס **משכורות**, באירוע **לפני עדכון** של הטופס הוסף את הקוד הבא שמטרתו למלא חלק מהמפתח שאינו מופיע בטופס משכורות באופן אוטומטי.

```
If IsOpen("עובדים") Then
  [מספר עובד] = Forms![עובדים]![קוד_עובד]
End If
```

בניית שאילתות בעזרת טופס

נניח שברצוננו להריץ שאילתות שיציגו את כל התשלומים שבוצעו מתאריך כלשהו לתאריך שני, ומסכום תשלום כלשהו לסכום תשלום אחר. נבנה טופס (שייקרא **טופס לשאילתה**) עם תיבות טקסט עבור גבול עליון ותחתון של תאריך ותשלום וכאשר האשף מכובה, ניצור לחצן פקודה עם הכיתוב **&הרצת שאילתה**. נקשר את התוכנית ללחצן באירוע **בעת לחיצה**:

```
Private Sub פקודה8_Click()
BuildSQL gstrWhere
End Sub
```

הסבר: השיגרה BuildSQL בונה שורת SQL על בסיס הנתונים שהוזנו ל**טופס לשאילתה**. את שורת SQL נחזיר במחרוזת גלובלית gstrWhere, המוגדרת בהצהרות מודול כלשהו באמצעות המילה Public במקום Dim. השיגרה BuildSQL נראית כך:

```
Public Sub BuildSQL(strSQL As String)
Dim strWhere As String
If Not IsNull(datPayFrom) Then
  strWhere = strWhere & " AND תאריך_תשלום >= #" &  datPayFrom & "#"
End If
If Not IsNull(datPayTo) Then
  strWhere = strWhere & " AND תאריך_תשלום <= #" & datPayTo & "#"
End If
If Not IsNull(curPayfrom) Then
  strWhere = strWhere & " AND סכום_תשלום >= " & curPayfrom
End If
If Not IsNull(curPayTo) Then
  strWhere = strWhere & " AND סכום_תשלום <= " & curPayTo
End If
If Len(strWhere) > 6 Then
  strSQL = Mid$(strWhere, 6)
End If
' MsgBox strSQL
End Sub
```

הסבר: הרעיון הכללי הוא שאנו בונים את המחרוזת strWhere על ידי הוספת הביטויים שנובעים מההגבלות שהוזנו לטופס. כל אחד מהם מתחיל במילה AND משום שייתכן שהוא מחובר לביטוי שלפניו. אנו בודקים כל אחד כדי לראות שאינו שווה ל-Null, משום שייתכן שהמשתמש לא הזין ערך בתיבת טקסט מסוימת. אם אכן כך, איננו רוצים להוסיף Null למחרוזת SQL. לאחר הרכבת המחרוזת קיימות שתי אפשרויות: או שהמחרוזת ריקה, או שיש משהו במחרוזת והמילה הראשונה במחרוזת היא AND. את המקרה השני נגלה על ידי בדיקת אורך המחרוזת strWhere. אם אורכה יותר מ-6, נוריד את 5 המקומות הראשונים. המחרוזת מוכנה לשימוש ומוחזרת לשיגרה הקוראת.

עכשיו כשיצרנו מחרוזת ב-SQL, השאלה היא כיצד להשתמש בה. ראשית, יש צורך ליצור טופס לקליטה והצגת תשלומים. נבנה טופס אוטומטי: עמודות, ונקרא לו Tashlumim.

כעת קיימות 3 דרכים והן:

בטופס Tashlumim, נדביק לאירוע **בעת סינון** (On Filter) את הקוד הבא:

```
Private Sub Form_Filter(Cancel As Integer, FilterType As Integer)
On Error GoTo Form_Filter_Error
gstrWhere = ""
' Display טופס לשאילתה
DoCmd.OpenForm "טופס לשאילתה", , , , , acDialog

' Set the forms Filter property to gstrWHERE which is a global
' variable which has  been set by  טופס לשאילתה

Me.Filter = gstrWhere
' MsgBox "xx" & Me.Filter
' And apply the filter
Forms!Tashlumim.FilterOn = True
' Prevent the default Filter window from opening
Cancel = True
Form_Filter_Exit:
Exit Sub

Form_Filter_Error:
Select Case Err.Number
    Case Else
        MsgBox Err.Description
        Resume Form_Filter_Exit
End Select
End Sub
```

הסבר: ננקה את המחרוזת gstrWhere, ונפתח את הטופס **טופס לשאילתה**. הפרמטר **acDialog** גורם לכל החלונות הפתוחים האחרים להיות משותקים עד לסגירת החלון הנוכחי (טופס לשאילתה). אחר נקבע את ערך המסנן כ-gstrWhere, ונפעיל את המסנן החדש שהוא מחליף את חלון המסנן שנפתח אוטומטית בעת לחיצה על הלחצן **סנן לפי טופס**. נבטל מסנן זה.

שיטת העבודה החדשה היא לפתוח את הטופס תשלומים, ואם נרצה להפעיל את המסנן החדש, קרי **טופס לשאילתה**, נלחץ על הלחצן **סנן לפי טופס**, נקליד את הערך הרצוי לסינון ונלחץ על **אישור** כדי לקבל את הרשומות המסוננות.

גישה שנייה היא לפתוח את התהליך בפתיחת **טופס לשאילתה**, לבנות לחצן ולהדביק לו את הקוד דלהלן:

```
Private Sub 15פקודה_Click()
BuildSQL gstrWhere
DoCmd.OpenForm "Tashlumim", , , gstrWhere
End Sub
```

הסבר: הקוד פותח את הטופס Tashlumim מסונן לפי gstWhere. שיטה זו מתאימה אם בדרך כלל יש צורך בסינון, השיטה הראשונה מתאימה אם ברוב המקרים אין צורך בו.

השיטה השלישית היא דרך אחרת להגיע לתוצאת השיטה השנייה. פתח את **טופס לשאילתה** והדבק את הקוד הבא ללחצן:

```
Private Sub 16פקודה_Click()
BuildSQL gstrWhere
MakeQueryDef gstrWhere
DoCmd.OpenForm "Tashlumim2"
End Sub
```

השיגרה MakeQueryDef יוצרת שאילתה עליה מבוסס הטופס Tashlumim2 (כלומר השאילתה MakeQuery) ומשמשת כ**מקור שורה** בטופס **Tashlumim2**. השיגרה **My Query** נראית כך:

```
Function MakeQueryDef(strSQL As String) As Boolean
If strSQL = "" Then Exit Function
Dim db As Database
Dim qdf As QueryDef
Set db = CurrentDb()
DoCmd.DeleteObject acQuery, "My Query"
Set qdf = db.CreateQueryDef("My Query")
qdf.SQL = "Select * from תשלומים where " & strSQL
qdf.Close
MakeQueryDef = True
End Function
```

הסבר: לאחר מחיקת הגירסה הקודמת של השאילתה, נבנה שאילתה חדשה על בסיס מחרוזת SQL שבנינו. השוני היחיד בין שיטה זו והקודמת הוא שהשאילתה שנוצרה נשארת בשם: My Query.

נספח

התקליטור המצורף

הוצאת הוד-עמי לספרי מחשבים

שליטה מלאה במחשב שלך.

ת.ד. 6108 הרצליה 46160   טלפון: 09-9564716   פקס: 09-9571582

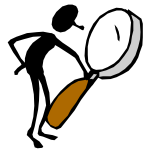
דואר אלקטרוני: info@hod-ami.co.il

בקרו אותנו באינטרנט: www.hod-ami.co.il

מה בתקליטור?

הוצאת הוד-עמי מגישה לך תקליטור הכולל את:

- **קטלוג CD** - קטלוג ספרי המחשבים הממוחשב האינטראקטיבי של **הוצאת הוד-עמי**. (נדרשת התקנת תוכנה. כולל חיפוש, הדפסה, דפדוף).
- **קטלוג HTML** - קטלוג ספרי המחשבים האינטראקטיבי של **הוצאת הוד-עמי**. (**לא** נדרשת התקנת תוכנה. מומלץ לצפייה עם Internet Explorer מגירסה 4 ומעלה).
- **ספרים לדוגמה** בהם ניתן לעיין וגם להדפיס.
- תוכנת **מבחן אישי** - לבדיקת ידע בתוכנות Word 7/97 ובה מאגר של מעל 400 שאלות.
- גיליון מלא של הירחון **חושבים חלונות** ובו מאמרים בנושאי Windows 95, Word 97 ו-Excel 97.
- מספר תוכנות עזר שימושיות.
- קבצי תרגול.



הערה חשובה:

אם מנהל התקן כונן התקליטורים הוא 16 סיביות - ייתכן ותראה רק 8 תווים ראשונים של שם הקובץ (במקרה ובמקור הוא ארוך יותר).

הסיבה: כונני תקליטורים במהירות x4 עובדים עם מנהל התקן שעבד בסביבת DOS ו-Windows 3.11 ויכול לעבוד גם עם Windows 95 למעט היכולת לזהות קבצים עם שמות ארוכים.

הפתרון: להתקין מנהל התקן 32 סיביות (אם קיים) או לקנות כונן תקליטורים חדש ולוודא שמצורף אליו מנהל התקן 32 סיביות.

קטלוג CD - הקטלוג הצבעוני האינטראקטיבי של הוד-עמי

הוצאת **הוד-עמי** גאה לבשר על הקטלוג **הצבעוני** האינטראקטיבי היחידי בארץ בתוכנה!

הקטלוג הצבעוני האינטראקטיבי **של הוד-עמי** עושה שימוש בטכנולוגיית **IBL** (**I**nternet/Intranet **B**ased **L**earning) שהינה המילה האחרונה בכל הקשור לשימושים ואפשרויות חדשות אשר גלומות בשימוש בטכנולוגיות אינטרנט.

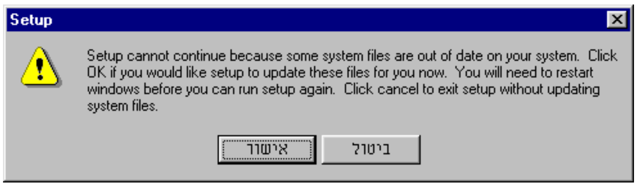
בעזרת **הקטלוג הצבעוני** האינטראקטיבי **של הוד-עמי** תוכל:

1. לעיין במידע על ספרי ההוצאה בכל עת שתרצה (לאחר התקנת התוכנה, בחר **התחל**, **תוכניות**, **קטלוג הוד-עמי**).
2. לעבור במהירות ובקלות בין **הקטלוג הצבעוני** האינטראקטיבי **של הוד-עמי** והיישום בו אתה עובד.
3. לדפדף בקטלוג **הצבעוני** האינטראקטיבי ספר אחר ספר, או לפי נושאים וקבוצות.
4. להדפיס את המחירון המלא ומידע על כל ספר.
5. לגשת במהירות, בגישה אינטואיטיבית, תוך התמקדות מהירה בספר המבוקש.
6. לעיין בקטלוג **הצבעוני** האינטראקטיבי בקצב אישי שלך.
7. לנווט את דרכך בקטלוג **הצבעוני** האינטראקטיבי ולחזור ולהתרענן בכל נושא בכל רגע.
8. להוריד עדכונים מהאינטרנט בכתובת **http://www.hod-ami.co.il/** ולהתעדכן בספרים חדשים.

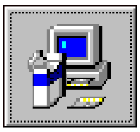
הקטלוג מתאים לעבודה ב-Windows 95 עם Internet Explorer 3.02a בגירסה העברית, אם עם זאת ניתן לצפייה עם דפדפן MS-IE4/5 (אך עם שיבושים קלים).

הקטלוג מתאים גם לעבודה ב-Windows 98 אם כי ההתקנה דורשת התערבות מצידכם כדי שתושלם. לפרטים קרא בקובץ README הנמצא בתיקיה X:\Catalog.

1. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
2. בתיבת הטקסט הקלד את הפקודה **X:\Catalog\Hod-ami** (החלף את האות X באות המייצגת את כונן התקליטורים שלך), ולחץ על **אישור**.
3. עקוב אחר ההוראות המופיעות על המסך ופעל לפיהן. כאשר מופיעה תיבת ההודעה **Setup** המודיעה על התקנת רכיבים חדשים - לחץ **אישור**. פעולה זו מצריכה אתחול מחדש של המחשב.



4. לאחר שהמחשב עולה מחדש הפעל את תוכנית ההתקנה מחדש:
   - לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
   - בתיבת הטקסט הקלד את הפקודה **X:\Catalog\Hod-ami** (החלף את האות X באות המייצגת את כונן התקליטורים שלך) ולחץ על **אישור**.
5. לחץ על לחצן **Setup**. מופיע חלון עם רקע כחול ובו הכותרת: "התקנת קטלוג הוד-עמי". לחץ **המשך**.
6. בחלון **התקנת קטלוג הוד-עמי** לחץ על תמונת המחשב. אם מופיעה תיבת דו-שיח בה תתבקש להגדיר היכן להתקין את Internet Explorer. השתמש בברירת המחדל. אל תיבהל, זו אינה התקנה "אמיתית" של Internet Explorer, אלא רק של מספר רכיבים הדרושים להפעלת הקטלוג (שים לב, שאין צורך בהתקנה מלאה של Internet Explorer). לחץ **OK**.



7. בסיום ההתקנה כולה תופיע תיבת דו-שיח. לחץ על **אישור**. יופיע חלון **Setup**, אשר את אתחול המחשב פעם נוספת על ידי בחירה בלחצן **כן**.
8. בחר בתפריט **התחל** באפשרות **הפעלה**.
9. בתיבת הטקסט הקלד **X:\Catalog\Update9809.exe** (החלף את האות X באות הכונן מתאימה), ולחץ על **אישור**.

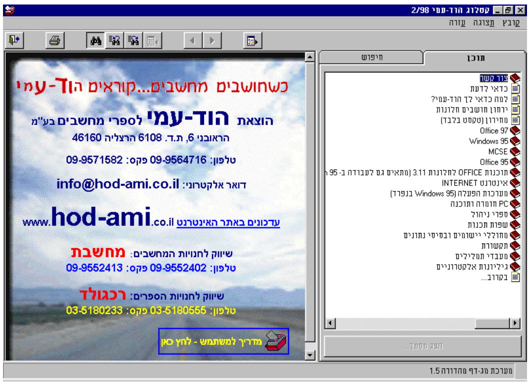
   זהו קובץ הנפרש אוטומטית וגם מותקן אוטומטית.
10. קרא את ההודעה ולחץ **אישור**.
11. בחלון WinZip Self-Extractor בחר בלחצן **Unzip** והמתן.

12. קרא את ההודעה ולחץ **אישור**.

13. בחלון WinZip Self-Extractor בחר בלחצן **Close**.

14. חזור על סעיפים 8-13 עבור קובץ **X:\Catalog\Update????.exe**. (אראבעת סימני השאלה הם עבור 9903 או מספר גדול יותר כמו 9904, 9905 וכדומה).

15. לאחר שהמחשב עולה מחדש לחץ על לחצן **התחל**, **תוכניות** ובחר **קטלוג הוד-עמי**.

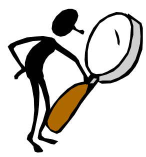


בכל חודש הקטלוג מתעדכן. ניתן להוריד קובץ עדכון מאתר ההוצאה בכתובת:

www.**hod-ami**.co.il

הפעלת התוכנה שהותקנה

להפעלת הקטלוג הצבעוני של הוצאת **הוד-עמי**, לחץ על **התחל**, עבור ל**תוכניות** ובתחתית התפריט תראה את **קטלוג הוד-עמי**. בחר בו.



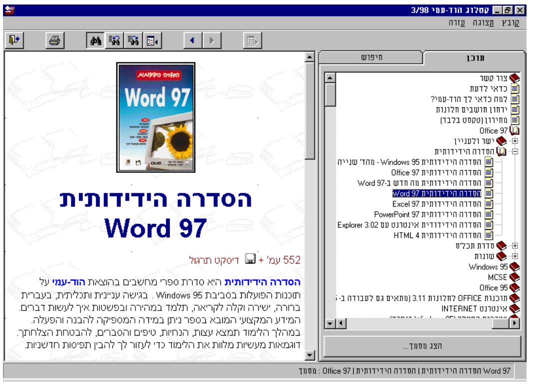
הערה:

התקנת הקטלוג ב-Windows 95 עשויה (לא תמיד) לשבש את העבודה עם הכתבן. במקרה זה יש להעתיק את קובץ Riched32.dll לתיקיית המערכת של Windows 95. בדרך כלל זו תיקיה C:\Windows\System

מבנה המסך הראשי

בהפעלת התוכנה יופיע מסך ובו שלושה אזורים עיקריים:

- מימין יופיע **סייר המסמכים** (דומה ל**סייר Windows**). וזהו למעשה עץ המציג את פרטי הקטלוג השונים. אזור זה ישמש בהמשך גם לתהליך החיפוש.
- בצד שמאל יופיע דף הפתיחה - וזהו **אזור התוכן**.
- בחלקן העליון יופיעו שורת התפריט וסרגל הכלים.



כך נראה מסך של קטלוג **הוד-עמי** הנמצא בתקליטור המצורף לספר

סיור מודרך בצעדים קלים

1. לחץ לחיצה כפולה על הסמל שלידו רשום **Office 97**.
2. לחץ לחיצה כפולה על הסמל שלידו רשום **הסדרה הידידותית/קורסים יעודיים**.
3. תיפתח רשימת מסמכים הקשורים לנושא.
4. לחץ לחיצה כפולה על הסמל שלידו רשום **קורסים יעודיים Word 97**.
5. בצד שמאל יוצג מסמך שכותרתו **קורסים יעודיים Word 97**.
6. לחץ על הסמל שבסרגל הכלים שבראש החלון כדי לעבור לדף הבא.
7. יופיע דף שכותרתו **הסדרה הידידותית הכרת המחשב האישי**.
8. גרור מטה את פס הגלילה של החלון השמאלי, כדי לקרוא את המשך התיאור.
9. אם ברשותך מדפסת, תוכל להדפיס את הדף על ידי לחיצה על הסמל .

שינוי מבנה המסך

ניתן לשנות את תצוגת המסמך על ידי שני לחצנים:

על ידי הצבת הסמן בין אזור סייר המסמכים לאזור המסמך יוצג חץ דו-ראשי. על ידי גרירה ימינה ושמאלה תוכל לשנות את חלוקת המסך בין שני האזורים.

הצג/הסתר סייר מסמכים. בלחיצה על לחצן זה, אזור המסמך "ישתלט" על כל המסך. לחיצה נוספת על לחצן זה תחזיר את המצב לקדמותו.

סרגל הכלים

	יציאה מהתוכנה.
	הדפס מסמך. המסמך המוצג על המסך יישלח להדפסה.
	הצג/הסתר סייר מסמכים. בלחיצה על לחצן זה, אזור המסמך "ישתלט" על כל המסך. לחיצה נוספת על לחצן זה תחזיר את המצב לקדמותו.
	הקודם/הבא בסייר המסמכים. בהתאם למיקום הסמן יוצג המסמך הקודם/הבא בעץ המסמכים (ראה פירוט בהמשך).
	הצג מיקום המסמך בתוכן. הסמן יתמקם בסייר המסמכים על השורה המתאימה למסמך הנוכחי. שים-לב, ייתכן שהסמן בעץ המסמכים ניצב על מסמך שאינו המסמך המופיע באזור התוכן באותו שלב (כדי להתאימם נשתמש בכלי הבא).
	מסמך קודם/הבא. דפדוף קדימה ואחורה במסמכים שכבר עברנו עליהם בהפעלה נוכחית (ראה פירוט בהמשך).
	הצג תת-עץ במסמך. כאשר הסמן ניצב על נושא המכיל מסמכים ו/או תת-נושאים נוספים, בשימוש בלחצן זה באזור סייר המסמכים ייפרסו כל תת-הנושאים והמסמכים, ובצד השני יוצג מסמך אחד ארוך אשר תוכנו מכיל את כל המסמכים.

דפדוף בעץ המסמכים

ניתן לבצע את פעולת הדפדוף ב- 3 דרכים שונות:

- מעבר על עץ המסמכים.
- פתיחת נושא.
- סגירת פרק.

מעבר על עץ המסמכים

- נושא כללי.
- נושא או תת-נושא פתוח.
- ספר.

פתיחת נושא

יש שלוש אפשרות לפתיחת נושא:

- לחיצה כפולה על שם הנושא.
- לחיצה על סימן "+" המופיע משמאל לסמל הנושא.
- על ידי מקש החיצים: →

סגירת פרק

יש שלוש אפשרות לסגירת פרק:

- לחיצה כפולה על שם הנושא.
- לחיצה על סימן "-" המופיע משמאל לסמל הנושא.
- על ידי מקש החיצים: ←

מעבר בין ענפי עץ המסמכים בעזרת לוח המקשים

- לחיצה על אות כלשהי תעביר את הסמן לשורה הבאה בעץ המתחילה באותה אות (אם אין שורה כזו, לא יקרה דבר).
- לחיצה על ↓ תעביר את הסמן לשורה הבאה ולחיצה על ↑ תעביר את הסמן לשורה הקודמת בעץ.

הצגת מסמך

יש שתי אפשרות להצגת מסמך:

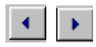
- כאשר הסמן ניצב על שם המסמך - לחיצה על לחצן 

.
- לחיצה כפולה על שם המסמך בעץ.

שימוש בלחצנים   בסרגל הכלים

דפדוף בעזרת לחצנים אלה דומה לדפדוף בספר, דף אחר דף, קדימה או אחורה. לחצנים אלה משמשים למעבר בין המסמכים, קדימה ואחורה, על פי סדר הופעתם בעץ התפריטים (נקודת ההתחלה היא מיקום הסמן בעץ). אם מגיעים לפרק או תת-נושא סגור - הוא ייפתח אוטומטית ויוצג המסמך הראשון.

כל לחיצה תפתח את המסמך המתאים (בחלק השמאלי) ותעביר את הסמן לשורה המתאימה בעץ (בחלק הימני).

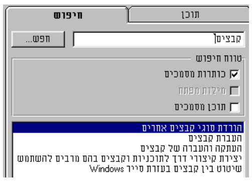
שימוש בלחצנים  בסרגל הכלים

דפדוף בין מסמכים שהופעלו בהפעלה הנוכחית. המסמך הנוכחי הינו המסמך האחרון ממנו אפשר לעבור אחורה עד למסמך הראשון של ההפעלה (מסך הפתיחה). בדרך בין המסמך הראשון למסמך הנוכחי תוכל לדפדף קדימה ואחורה.

הורדת עדכונים

הקטלוג הצבעוני מתעדכן מדי חודש. את העדכון ניתן להוריד מהאינטרנט (חינם!) מאתר ההוצאה. בפתיחת הקטלוג בצד שמאל כתוב **עדכונים באתר האינטרנט**. אם אתה מחובר לאינטרנט, הצב את הסמן על כתובת זו ולחץ.

להרחבת התצוגה לחץ על **משקפת**. באתר **הוד-עמי** בחר בקישור **קטלוג והורדת עדכון חודש**. הורד את עדכון הקטלוג (סבלנות, ייתכן שיעברו מספר דקות) וזכור היכן אתה שומר את הקובץ. צא מהקטלוג ונתק את החיבור לאינטרנט. הפעל את **סייר Windows** ואתר את הקובץ. לחץ לחיצה כפולה על הקובץ. קרא את ההודעה ולחץ על **אישור**. לחץ **UnZip**, קרא את ההודעה ולחץ **אישור**. לחץ **Close** והפעל מחדש את הקטלוג. זהו, הקטלוג מעודכן!!!



חיפוש

עבור לכרטיסיה **חיפוש** בחלק הימני של החלון.

1. הקלד רצף תווים. זו יכולה להיות מילה או חלק ממילה.
2. קבע את טווח החיפוש - האם לחפש בכותרות המסמכים ו/או בתוכן המסמכים.
3. הקש Enter או לחץ על . אם נמצאו מסמכים, תופענה כותרותיהם.

הצגת המסמך

יש שתי אפשרויות להצגת מסמך:

- כאשר הסמן ניצב על שם המסמך - לחיצה על לחצן .
- לחיצה כפולה על שם המסמך.

שים לב! במסמך שמוצג מחלון החיפוש, הטקסט שחיפשת יודגש בצבע אדום.

העתקת טקסט (ללא תמונות) לתוכנות אחרות

אפשר לסמן קטע ממסמך, להעתיקו ולהדביקו במקום אחר (למשל, במסמך Word).

1. סמן את הטקסט שברצונך להעתיק (הסימון מתבצע על ידי לחיצה וגרירת העכבר על האזור שברצונך להעתיק). הטקסט שסומן ייצבע בצבע כחול.

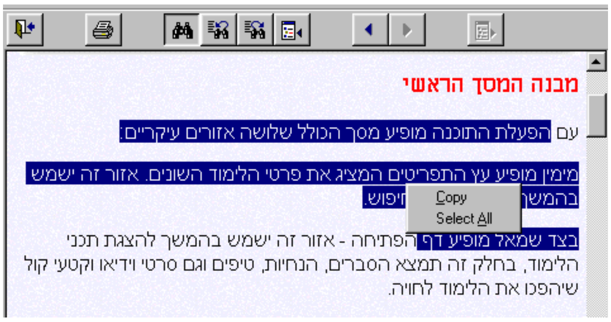


הערה:
לבחירה וסימון כל הטקסט בחלון השמאלי:

1. הצב את סמן העכבר באזור התוכן.
2. לחץ לחיצה ימנית.
3. בחר **Select All**.

2. ביצוע העתקה:

- הצב את סמן העכבר על הקטע המסומן.
- לחץ לחיצה ימנית ובחר **Copy**, או לחץ **Ctrl+C**.



- עבור לתוכנה אחרת, כמו Word, פנקס רשימות, כתבן וכדומה.
- הצב את נקודת הכניסה במקום שבו תרצה להוסיף את הטקסט ובצע אחת מהפעולות הבאות: בחר בתפריט **עריכה**, **הדבק**, או לחץ על לחצן **הדבק**, או הקש **Ctrl+V**, או לחץ לחיצה ימנית ובתפריט המקוצר בחר **הדבק**.

העתקת תמונה לתוכנות אחרות

אפשר להעתיק תמונה ממסמך שמופיע על המסך ולהדביקה במקום אחר (למשל במסמך Word).

1. הצב את הסמן על התמונה שברצונך להעתיק.
2. לחץ לחיצה ימנית ובתפריט המקוצר בחר **Copy**.
3. עבור לתוכנה אחרת: תוכנה גרפית, Word וכדומה.

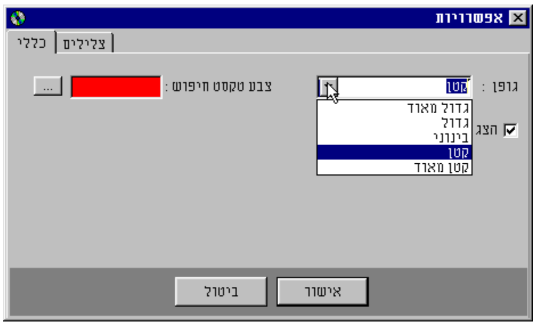
4. לביצוע ההדבקה,

- בחר בתפריט **עריכה**, **הדבקה**
- או לחץ על לחצן **הדבקה**
- או הקש **Ctrl+V**
- או לחץ לחיצה ימנית ובתפריט המקוצר בחר **הדבקה**.

אפשרויות תצוגה

בשורת התפריט בחר **תצוגה**, **אפשרויות.** בחלון זה ניתן לשנות את המאפיינים האלה:

- גודל הגופן המוצג במסמכים השונים.
- צבע טקסט החיפוש (ברירת המחדל - אדום).
- האפשרות להציג שורת מצב או לא.
- הצלילים שמשמיעה התוכנה כאשר מתחלפים המסמכים.



בתיבת הדו-שיח **אפשרויות** תוכל לקבוע גם את צבע טקסט החיפוש

קטלוג HTML - קטלוג צבעוני

הוצאת הוד-עמי גאה לבשר על קטלוג HTML **צבעוני** העושה שימוש בטכנולוגיית DHTML (Dynamic HTML) שהינה המילה האחרונה בעיצוב דפי Web באינטרנט.

ניתן לצפייה באמצעות Microsoft Internet Explorer מגירסה 4 ומעלה.

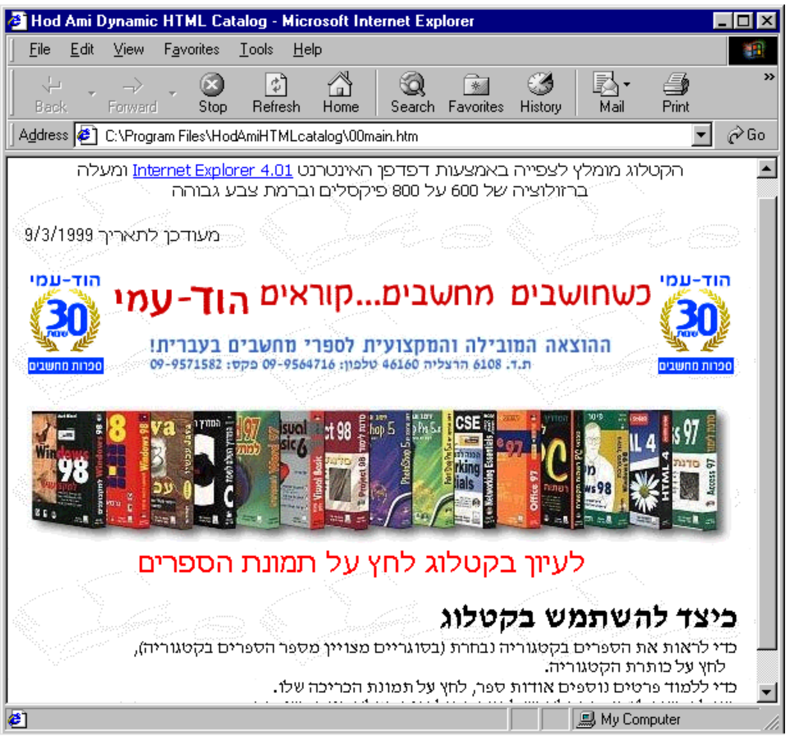
בעזרת קטלוג HTML **צבעוני** תוכל:

- לעיין במידע על ספרי ההוצאה בכל עת שתרצה (לחיצה כפולה.... וזהו!).
- לעבור במהירות ובקלות בין הקטלוג **הצבעוני** והיישום בו אתה עובד.
- לעיין במידע על כל ספר וספר.
- לגשת במהירות, בגישה אינטואיטיבית, תוך התמקדות מהירה בספר המבוקש.

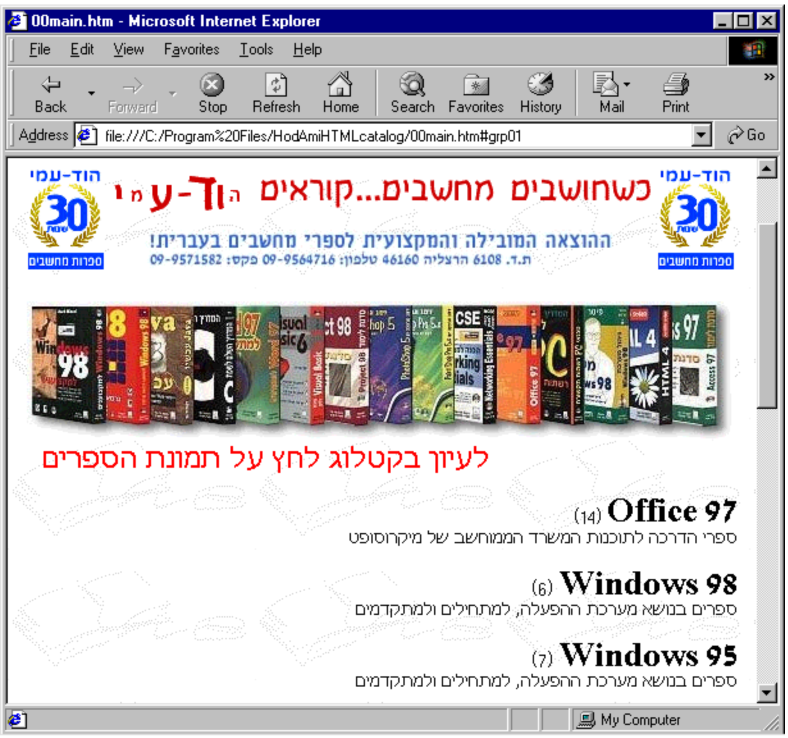
- לעיין בקטלוג **הצבעוני** בקצב אישי שלך.
- לנווט את דרכך בקטלוג **הצבעוני** ולחזור ולהתרענן בכל נושא בכל רגע.

הקטלוג ניתן לצפייה באמצעות Internet Explorer מגירסה 4 ומעלה.

1. פתח את סייר Windows.
2. עבור לתיקיה HTML Catalog אשר בתקליטור המצורף.
3. אתר את הקובץ 00main.htm והפעל אותו (בדרך כלל על ידי לחיצה כפולה).



4. קרא את ההוראות **כיצד להשתמש בקטלוג** שבמסך פתיחה זה.
5. לחץ על תמונת הספרים.
6. יוצגו הנושאים השונים: Office 97, Windows 98 וכדומה.



7. הצב את הסמן על כותרת הנושא, למשל, **Windows 98** ולחץ בעכבר.

8. יוצגו הספרים השייכים לאותו נושא.



9. לחץ בעכבר על הספר לגביו אתה מעוניין במידע (שים לב, הסמן **לא** ישתנה).

10. ייפתח חלון בו תוכל לקרוא מידע אודות הספר (מה הוא מכיל, למי הוא מיועד, מה נמצא בתקליטור המצורף, מיהו המחבר, כמה עמודים בספר וכדומה).

שים לב - כדי להמשיך ולעיין בקטלוג עליך **לסגור** את החלון המכיל את פרטי הספר.

התקנת קטלוג HTML

מי שרוצה להפעיל את קטלוג HTML מהתקליטור, צריך לנהוג לפי ההוראות הרשומות לעיל. הפעלה מהתקליטור מחייבת את הימצאות התקליטור בכונן. בהתקנה פשוטה ניתן להעתיק את הקטלוג לדיסק הקשיח. להלן הוראות להתקנת קטלוג HTML:

1. לחץ על **התחל** ובחר **הפעלה**.
2. בעזרת לחצן **עיון** סמן את הקובץ **HodAmiHTMLcatalog.EXE** אשר בתיקיה **X:\HTML Catalog**.
3. לחץ **פתח**.
4. לחץ **אישור**.
5. לחץ **אישור**.
6. לחץ על לחצן **Unzip**
7. לחץ **אישור**.
8. על שולחן העבודה מופיע סמל עם כיתוב **קטלוג ספרי מחשבים בהוצאת הוד-עמי**.
9. הפעל אותו.



התקנה - Acrobat Reader

תוכנה זו יש להתקין כדי לקרוא את דוגמת הירחון "**חושבים חלונות**" ואת **הספרים לדוגמה** וכדי לפתוח כל קובץ שהסיומת שלו pdf. התוכנה פועלת במערכות הפעלה **Windows 95/98 בלבד!**

1. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
2. בתיבת הטקסט הקלד את הפקודה **X:\Software\Adobe Acrobat\Ar32e301** (החלף את האות X באות המייצגת את כונן התקליטורים שלך) ולחץ על **אישור**.
3. בחלון הבא בחר בלחצן **כן** להמשך ההתקנה.
4. אשף ההתקנה מתקין את הרכיבים הנדרשים. עליך ללחוץ על **Next**, **Yes** ו-**Next** פעם נוספת כדי לסיים את ההתקנה.
5. בסיום ההתקנה לחץ על **Finish**.
6. קרא את הודעת סיום ההתקנה שמופיעה, ולחץ **אישור**.

ספרים לדוגמה

בתיקיה **Books DEMO** תמצא מספר דוגמאות מספרי ההוצאה. כדי לראות וגם להדפיס את הקבצים (פורמט pdf) יהיה עליך להתקין את Acrobat Reader (ראה לעיל).

1. פתח את תפריט **התחל**, **תוכניות**, **Adobe Acrobat**, **Acrobat Reader 3.01**.
2. פתח את תפריט **File** ובחר באפשרות **Open**.
3. בתיבה **חפש ב**: בחר בכונן התקליטורים שלך ועבור לתיקיה **Books DEMO**.
4. בחר בקובץ ולחץ על לחצן **Open**.

חושבים חלונות - הפעלה

1. פתח את תפריט **התחל**, **תוכניות**, **Adobe Acrobat**, **Acrobat Reader 3.01**.
2. פתח את תפריט **File** ובחר באפשרות **Open**.
3. בתיבה **חפש ב**: בחר בכונן התקליטורים שלך ועבור לתיקיה **ThinkWin**.
4. בחר בקובץ **ThinkWin - Vol 5.pdf** המופיע בתיקיה זו, כדי לקרוא את גיליון מספר 5 במלואו.
5. בחר בקובץ **ThinkWin - Brief.pdf** המופיע בתיקיה זו, כדי לקרוא את תקצירי הגליונות הקודמים.

ומה הלאה...

הזמנת מנוי - חושבים חלונות

או פקסס
09-9571582

שלח בדואר לפקודת **הוד-עמי ת.ד. 6108 הרצליה 46160**

סמן/י X לפי בחירתך:

אני מעוניין/ת לעשות מנוי על הירחון **חושבים חלונות**.

☐ 12 גיליונות במחיר 99₪ כולל מע"מ

חובה למלא פרטים

שם משפחה ____________ שם פרטי ____________

כתובת מלאה ____________

ישוב ____________

מיקוד ____________

טלפון ____________ פקס ____________

e-mail: ____________

ויזה - 16 ספרות דיינרס - 14 ספרות
ישראכרט - 8 ספרות מאסטרקארד - 16 ספרות

מספר כרטיס אשראי

בתוקף עד ____ / ____ תעודת זיהוי ____________

כתובת למשלוח החשבונית (במידה והיא שונה מהכתובת שלעיל):

____

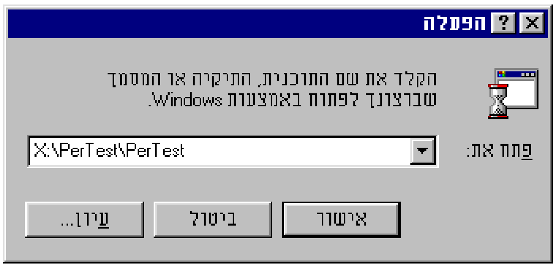
____

אישור: תאריך חתימת המזמין

מבחן אישי - התקנה

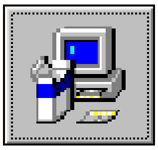
מבחן אישי מאפשר לך לבחון את ידיעותיך בתוכנות Office השונות. בתקליטור המצורף לספר זה תמצא מספר מבחנים לתוכנת Word בגרסאות 7 ו-97.

1. בחר בתפריט **התחל** באפשרות **הפעלה**.
2. בתיבת הטקסט הקלד **X:\PerTest\Pertest.exe** (החלף את האות X באות הכונן המתאימה), ולחץ על **אישור**.



זהו קובץ הנפרש אוטומטית וגם מותקן אוטומטית.

3. קרא את ההודעה ולחץ **אישור**.
4. בחלון WinZip Self-Extractor בחר בלחצן **Unzip** והמתן.
5. לחץ **אישור**.



תהליך ההתקנה מתחיל. המתן.

6. לחץ **OK**.
7. לחץ על תמונת המחשב. המתן.
8. מופיעה הודעה שההתקנה הסתיימה בהצלחה. לחץ **אישור**.

התקנת ערכת מבחנים מלאה ל-Word 7/97

1. בחר בתפריט **התחל** באפשרות **הפעלה**.
2. בתיבת הטקסט הקלד **X:\PerTest\PTWord.exe** (החלף את האות X באות הכונן המתאימה), ולחץ על **אישור**.
3. זהו קובץ הנפרש אוטומטית וגם מותקן אוטומטית.
4. קרא את ההודעה ולחץ **אישור**.
5. בחלון WinZip Self-Extractor בחר בלחצן **Unzip** והמתן.
6. קרא את ההודעה ולחץ **אישור**.
7. בחלון WinZip Self-Extractor בחר בלחצן **Close**.

הפעלה - מבחן אישי הוד-עמי

לחץ על לחצן **התחל**, **תוכניות**, **מבחן אישי - הוד עמי**.

מה עוד בתקליטור?

הוצאת **הוד-עמי** מפיצה תוכנות אלו כבונוס ללקוחות ההוצאה, ואינה מתיימרת לגבות תשלום עבור התוכניות המצורפות ו/או לתמוך בהם.



אזהרה:

השימוש בתקליטור זה הוא על אחריותו הבלעדית של המשתמש. המוצרים המותקנים בתקליטור זה מסופקים באחריות החברות המייצרות אותם. הוצאת **הוד-עמי** אינה אחראית, בכל צורה שהיא, לאופן ולטיב התוכנות המותקנות.

בכל שאלה לגבי תוכנה הנמצאת בתקליטור, יש לפנות למפתחי התוכנה (כל תוכנה בנפרד) כפי שמצוין בקבצי העזרה של התוכנה המדוברת.

הקבצים הם גרסאות **שיתופיות** (ShareWare) ו**חופשיות** (FreeWare).

גירסה **שיתופית** (ShareWare) מאפשרת לך, המשתמש, לבדוק את יעילות התוכנה ואת תאימותה לעבודה אותה אתה מבצע. אם נמצאה התוכנה מתאימה לצרכיך, עליך לשלם למפתחיה תשלום סמלי (לפי הרשום בקבצי העזרה של כל תוכנה ותוכנה בנפרד) כדי לקבל רישיון מלא לשימוש בה. רכישת רישיון לשימוש בתוכנה יפתח בפניך מיגוון אפשרויות שייתכן ולא עמדו לרשותך בהפעלת הגירסה השיתופית.

FontsPekan

קובץ זה יתקין במחשב 2 גופנים בעברית לשימושכם. בסיום ההתקנה יש לבצע את הפעולות הבאות:

1. לחץ על **התחל**, הצבע על **הגדרות**, ובחר ב**לוח הבקרה**.
2. לחץ לחיצה כפולה על הסמל **גופנים**.
3. פתח את תפריט **קובץ** ובחר באפשרות **התקנת גופן חדש**.
4. עבור לתיקיה C:\FontsPekan.
5. לחץ על לחצן **בחר הכל** (סה"כ יש בתיקיה 2 גופנים).
6. ודא שתיבת הסימון **העתק גופנים לתיקיית הגופנים** מסומנת.
7. לחץ **אישור**.
8. סגור את חלון התיקיה **Fonts**.
9. סגור את חלון **לוח הבקרה**.

כעת, מוכנים הגופנים לשימוש בכל התוכנות המותקנות במחשב שלך: Word, Excel, PowerPoint וגם בתוכנות גרפיות, כגון Paint Shop Pro ו-PhotoShop.

הגופנים נקראים Tml-JUMP ו-Tml-step ויופיעו בתחתית רשימת שמות הגופנים (בדרך כלל). הרי דוגמה שלהם:

Tml-step

אבגדהוזחטיכךלמםנןסעפףצץקרשת1234567890

Tml-JUMP

אבגדהוזחטיכךלמםנןסעפףצץקרשת1234567890

Terra - מימד חדש בתצוגה
ים המלח ממעוף הציפור

Terra היא משפחת מוצרים של חברת **סקייליין מערכות תוכנה**, הבנויה סביב מנוע גרפי ייחודי ורב עוצמה, המאפשר תצוגה בזמן אמת של פני שטח בתלת-מימד ללא הגבלת פרטים וגודל על גבי מחשבים ביתיים. מקור תמונת השטח הוא בתצלומי לווין ותצלומי אוויר.

דרישות מערכת מינימליות:

- מעבד מסוג Pentium בטכנולוגיית MMX
- זיכרון פנימי 32MB RAM
- זיכרון כרטיס מסך 2MB
- כונן תקליטורים במהירות x6
- מערכת הפעלה Windows 95/98
- התקליטור חייב להיות בכונן במהלך ההפעלה

מנוע Terra לוקח את התצוגה בזמן אמת לרמות חדשות שלא נודעו בעולם מחשבי ה-PC. הוא מאפשר אין סוף מצבי תצוגה: החל ממבט קרוב לפרטי פרטים, ועד תצוגה רחוקה עד האופק. מנוע Terra עושה שימוש בטכנולוגיית MMX החדשנית, ומאפשר איכויות שבעבר היו נחלתם של מחשבים יקרים בלבד.

לפרטים נוספים על התוכנה ומפתחיה קרא בקובץ INFO בתיקיה Terra.

התקנת Terra

מומלץ להסיר גירסה קודמת של Terra, אם קיימת.

1. הכנס את התקליטור לכונן.
2. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
3. לחץ על לחצן **עיון**.
4. בחר בכונן התקליטורים בתיקיה **Terra** ובקובץ בשם **SETUP.EXE**.
5. לחץ על לחצן **פתח**.
6. לחץ על לחצן **אישור**.
7. פעל לפי ההוראות על המסך לפי הסדר (מימין לשמאל):
   Next, Yes, Next, Next, Next, Next, Finish.

ייתכן ובמהלך ההתקנה תתבקש להתקין רכיבי DirectX. אם במחשב שלך מותקנת מערכת ההפעלה Windows 95 הפועלת עם ממשק עברית (לחצן **התחל**) יהיה עליך לשנות את ההגדרות האזוריות:

1. בחר בלחצן **התחל**, **הגדרות**, **לוח הבקרה**, **הגדרות אזוריות**.
2. במקום **עברית** בחר **אנגלית** (ארצות הברית).
3. בחר בלחצן **החל**.
4. לשאלה האם ברצונך לאתחל את המחשב מחדש? ענה **כן**.
5. עתה, יהיה עליך להתחיל את התקנת Terra מחדש.

הפעלת Terra

לחץ על לחצן **התחל**, **תוכניות**, **Terra**, **TerraViewer**.

פעולה	מקשים
הגבר מהירות	Shift+A
האט מהירות	Shift+Z
תנועה מעלה/מטה/ימינה/שמאלה	מקשי חיצים

הוראות הפעלה מפורטות נמצאות בתפריט **Help** שבתוכנת Terra.

התקליטור חייב להיות בכונן בעת הפעלת התוכנה.



טיפ:

תוכל להאט את המהירות בעזרת Shift+Z לא רק למהירות אפס, אלא מתחת לזה. המשמעות היא ש... תטוס אחורה!!! שווה בדיקה!!!

התקנת תוכנת גלישה לאינטרנט (בגירסה העברית) Microsoft Internet Explorer 4.01

מומלץ להסיר גירסה קודמת של Internet Explorer, אם קיימת.

1. הכנס את התקליטור לכונן.
2. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
3. לחץ על לחצן **עיון**.
4. בחר בכונן התקליטורים בתיקיה Software\IE401sr1\I386 ובקובץ בשם Setup.exe.
5. לחץ על לחצן **פתח**. לחץ על לחצן **אישור**.
6. פעל לפי ההוראות על המסך.

התקנת תוכנת Microsoft Internet Explorer 5

בגרסת Enabled, רואים עברית, תפריטים באנגלית. התקנת IE5 מחייבת התקנה קודמת של דפדפן כלשהו מבית Microsoft.



אזהרה:
גירסה זו **אינה** מיועדת להתקנה במחשב בו מותקנת מערכת ההפעלה Windows 98 **בעברית** (זו שבה התפריטים הם בעברית).

1. הכנס את התקליטור לכונן.
2. לחץ על לחצן **התחל** ובחר באפשרות **הפעלה**.
3. לחץ על לחצן **עיון**.
4. בחר בכונן התקליטורים בתיקיה Software\IE5 ובקובץ בשם Setup.exe.
5. לחץ על לחצן **פתח**.
6. לחץ על לחצן **אישור**.
7. פעל לפי ההוראות על המסך.

היכן נמצאים הקבצים הקשורים לספר זה?

תחת התיקיה **Books** תוכל למצוא את התיקיה הרלוונטית לספר שאליו מצורף התקליטור. שם התיקיה ***59236***.

העתק את שני הקבצים לתיקיה בדיסק הקשיח.

אנו מבקשים להתייחס לקבצים הנמצאים בתיקיה הרלוונטית לספר, שאליו מצורף תקליטור זה, **<u>בלבד</u>**. בתיקיות אחרות נמצאים קבצים הרלוונטיים לספרים אחרים.

שם התיקיה המכילה את הקבצים בנוי מה-**דאנאקוד** הנמצא בעטיפה האחורית של הספר. שם התיקיה בנוי מ-5 ספרות לפי התרשים הבא:



תיקיה ראשית SoftWare (רשימה חלקית)

שם תוכנה	תיאור	קובץ הפעלה	מבצע
Adobe Acrobat	תוכנה לצפייה בקבצי pdf	Ar32e301.exe	התקנה
Babylon	תוכנה לתרגום/בדיקת איות	B30502h3.exe	התקנה
Clean System	מחיקת קבצי dll שאין צורך בהם (מאמר בנושא הופיע ב**חושבים חלונות**, גיליון 22)	ClnSys16.exe	התקנה
FontsPekan	גופנים בעברית	FontsPekan.exe	פריסה
Gohst 5.1c	תוכנה להעתקת תוכן כוננים קשיחים	g51c_trl.exe	פריסה
ICQ	תוכנה לתקשורת אישית באינטרנט	ICQ99a.exe	התקנה
Paint Shop Pro 5.03	תוכנה ליצירת, עיצוב ועיבוד תמונות	Psp503ev.exe	התקנה
Power Toys	תוכניות שירות עבור Windows 95	PowerToy.exe	פריסה
WinAmp	תוכנה להשמעת קבצי MP3 (מוסיקה)	WinAmp.exe	התקנה
WinZip	תוכנית לפריסה/דחיסה של קבצים	WinZip7sr1.exe	התקנה
WordView	תוכנית לצפייה בקבצי doc	WordView.exe	התקנה

אינדקס

א

ב

ד

ה

ט

י

ל

מ

ת

Index

S

T